WERKSTATT UNTERNEHMENSSOFTWARE KARLSRUHE (WUSKAR)

Fallstudie

INTEROPERABLE ABSICHERUNG DES NACHRICHTENAUSTAUSCHS ZWISCHEN WEBSERVICES AM BEISPIEL DES TRANSCRIPT OF RECORDS-BPEL-SERVICES

Sebastian Kreuzer
Yining Zhao

Frank Brandt
Christian Emig
Sebastian Abeck

Cooperation & Management (C&M)
Institut für Telematik
Universität Karlsruhe (TH)
wuskar@cm-tm.uka.de

Kurzbeschreibung

Das vorliegende Dokument stellt die Ergebnisse des im Rahmen einer „WUSKAR-Fallstudie“ durchgeführten Team-Projektes vor. Der bereits vorhandene BPEL-Webservice „Transcript of Records“ verarbeitet sensible Daten und soll daher abgesichert werden. Aufgrund der Heterogenität der zugrunde liegenden Systeme und unter Berücksichtigung zukünftiger, denkbarer Integrationsvorhaben wird zur Gewährleistung der Interoperabilität der Absicherungslösung der OASIS-Standard WS-Security eingesetzt. Als Vorleistung für die Absicherung werden die Basis-Webservices zunächst auf eine WS-Security-fähige Ausführungsplattform portiert.

Schlüsselwörter

WS-Security, Absicherung, Webservices, JBoss, AXIS, WUSKAR, ToR, BPEL, Oracle, SAP CM, MySQL

Ziele

(1) Migrate Web Services from Apache Tomcat/AXIS to JBoss AS

(2) Securing the ToR Services with WS-Security
  (1) Secure the Core Web Services (JBoss AS)
  (2) Secure the Composite BPEL Service (Oracle BPEL PM)

Information 1: WUSKAR CASE STUDY - Goals

(1) Als Vorleistung für die Absicherung der ToR-Webservices wird die Umsiedlung der bestehenden Webservices von einer „Apache Tomcat/AXIS“-Umgebung auf den JBoss Application Server benötigt.

(2) Die zentrale Problemstellung besteht in der Absicherung des Nachrichtenaustausches zwischen Webservice-Providern und Nutzern durch den Einsatz von WS-Security. Konkret wird der Nachrichtenaustausch zwischen dem ToR-BPEL-Prozess und den beteiligten Basis-Webservices betrachtet.

Inhaltsverzeichnis

0 DAS WUSKAR-PROJEKT 4
0.1 Die „WUSKAR-Fallstudie“ - Allgemeine Zielsetzung 4
0.2 Die WUSKAR-Fallstudie „Transcript of Records“ 4
1 ANALYSE 7
1.1 Geschäftsbereich 7
1.1.1 Anwendungsfälle und beteiligte Rollen 7
1.1.2 Geschäftsobjekte 8
1.1.3 Geschäftsprozess 9
1.1.4 Sicherheitsanforderungen aus Geschäftssicht 10
1.2 Systembereich 11
1.2.1 Vorhandene Systeme 12
1.2.2 Darstellung der Geschäftsobjekte 12
1.2.3 Sicherheitskonzepte und Sicherheitstechnologien 13
1.2.4 Ausblick: WS-Security und verwandte Standards 16
1.2.5 Sicherheitsanforderungen aus Systemsicht 17
1.3 Übungsaufgaben 19
2 ENTWURF 20
2.1 Randbedingungen und Überblick 20
2.1.1 Signieren und Verschlüsseln mit WS-Security 20
2.1.2 Erweiterung um Zugriffskontrolle 22
2.2 Architektur auf Dienstkompositionsebene 24
2.2.1 Oracle SOA Suite 24
2.2.2 Oracle BPEL Process Manager 25
2.2.3 Oracle Web Services Manager 27
2.3 Architektur auf Ebene der Basis-Webservices und der angekoppelten Altsysteme 29
2.3.1 J2EE - Überblick 29
2.3.2 JBoss Appilcation Server 32
2.3.3 JBossWS - JBoss Web Services 38
2.3.4 JBossWS und WS-Security 38
2.3.5 Ankopplung von SAP CM 40
2.4 Übungsaufgaben 41
3 IMPLEMENTIERUNG 42
3.1 Migration der Basis-Webservices von AXIS auf den JBoss Application Server 42
3.1.1 Extraktion der Geschäftsfunktionalität aus den alten AXIS-Webservices 42
3.1.2 Entwicklung der neuen JBoss-Webservices 44
3.2 Absicherung der Webservice-Aufrufe mit dem Oracle Web Services Manager 46
3.2.1 Schlüsselverteilung 47
3.2.2 Oracle Web Services Manager 47
3.3 Absicherung der Basis-Webservices mit dem JBoss Application Server 52

3.3.1 JBossWS - Konfiguration 52
3.3.2 Schlüsselerzeugung 53
3.4 Übungsaufgaben 55
4 AUSBLICK 56
4.1 Interoperabilität 56
4.2 Schlüsselverteilung 57
4.3 Schnittstellenerweiterung 57
Lösungshinweise zu den Aufgaben 59
Abkürzungen und Glossar 62
Index 67
Informationsfolien 67
Literatur 68

0 DAS WUSKAR-PROJEKT

Bei der Werkstatt Unternehmenssoftware Karlsruhe (WUSKAR) handelt es sich um ein vom Land gefördertes Projekt mit einer geplanten Laufzeit von drei Jahren. Ziel von WUSKAR ist es, für Studierende eine Plattform zu schaffen, mit der sie lernen, Fragestellungen aus der Praxis mithilfe von Unternehmenssoftware umzusetzen [EA04].

0.1 Die „WUSKAR-Fallstudie“ - Allgemeine Zielsetzung

Im Rahmen des WUSKAR-Projektes werden ausgewählte Geschäftsprozesse im Bereich der Aus- und Weiterbildung unter Einsatz verschiedener kommerzieller Unternehmenssoftware-Produkte (HISLSF, SAP-Software Campus Management, CAS Campus-System) gezielt unterstützt. Schwerpunkte bilden die Aspekte der Modellierung beteiligter Prozesse sowie die hierbei zu berücksichtigenden Aspekte der Sicherheit und des Datenschutzes [WUSKAR-FS].

Das Projektziel soll erreicht werden, indem Studierenden Zugang zu Computersystemen gestattet wird, welche in einem definierten Anfangszustand sind und auf denen komplexe Aufgaben aus der industriellen Praxis gelöst werden können, und zwar durch

- Installation
- Konfiguration
- Integration

kommerzieller Software.

Die Art der Aufgabenstellung für Studierende kann dabei ganz unterschiedlich sein. Denkbar sind

- theoretisch-grundlegende Aufgaben zur Prozessintegration
- anwendungsorientierte Fragestellungen auf Basis eines soliden Grundlagenwissens
- das Nachvollziehen konkreter Fallbeispiele aus der industriellen Praxis.

Als modulare Struktur für diese Aufgabenstellungen wurde der Begriff der „Fallstudie“ gewählt. Im Rahmen einer solchen Fallstudie zu betrachtende, werden Software-unterstützte Prozesse ausgewählt, sowohl der Geschäftsbereich wie der Systembereich modelliert und die Zusammenhänge beider Bereiche herausgestellt [WUSKAR-P].

0.2 Die WUSKAR-Fallstudie „Transcript of Records“

Der betrachtete Geschäftsprozess stammt aus im Rahmen des so genannten Bologna-Prozesses erfolgten Maßnahmen zur Schaffung eines einheitlichen europäischen Hochschulraumes. Zur Vereinfachung von Studienaufenthalten im Ausland sollen die Notenauszüge, *Transcript of Records* (ToR) genannt, der Hochschulen vereinheitlicht werden. Bevor jedoch ein ToR erstellt werden kann, sind weitere Voraussetzungen zu beachten [EA04].

Die WUSKAR-Fallstdie „*Transcript of Records*“ blickt mittlerweile auf folgende Historie zurück:

[WUSKAR-P] WusKar Werkstatt Unternehmenssoftware Karlsruhe, Dezember 2003.

[WUSKAR-FS] Sebastian Abeck, Christian Mayerl, Oliver Mehl, Zoltán Nochta: Projekt WUSKAR, C&M (Prof. Abeck), Universität Karlsruhe (TH), 2004.

[EA04] Christian Emig, Sebastian Abeck: Werkstatt Unternehmenssoftware

Karlsruhe - Ein Beitrag zur prAXISorientierten Informatik-Ausbildung an Hochschulen, UNIKATH, Karlsruhe 2004.

[AE04] Sebastian Abeck, Christian Emig, Jochen Weisser: Fallstudie Transcript of Records, Universität Karlsruhe (TH), C&M (Prof. Abeck), 2004.

[RS+05] E. Reuther, T. Stiller, S. Tardif d'Hamonville, J. Weisser, C. Emig, S. Abeck: Geschäftsprozess- und Systemmodellierung von SAP Campus Management, Projekt „Werkstatt Unternehmenssoftware Karlsruhe (WUSKAR)", C&M (Prof. Abeck), Universität Karlsruhe (TH), 2005.

[We05] J. Weisser: University SOA – Building a Transcript of Records Service, Study Thesis, Project „Werkstatt Unternehmenssoftware Karlsruhe (WUSKAR)", C&M (Prof. Abeck), Universität Karlsruhe (TH), 2005.

[BB05] C. Beutler, S. Enge: Entwicklung von Core-Webservices zur Generierung eines Transcript of Records, Team-Praktikum, Projekt „Werkstatt Unternehmenssoftware Karlsruhe (WUSKAR)", C&M (Prof. Abeck), Universität Karlsruhe (TH), 2005.

[SS05] Heiko Schandua, Tomas Stiller: Case Study University SOA, Team Study Thesis, Project „Werkstatt Unternehmenssoftware Karlsruhe (WUSKAR)", C&M (Prof. Abeck), Universität Karlsruhe (TH), 2005.

(1) ANALYSIS
  (1) Business Area
  (2) Systems Area

(2) DESIGN
  (1) Architecture Overview
  (2) Oracle BPEL PM
  (3) JBoss AS

(3) IMPLEMENTATION
  (1) Migrating Web Services from AXIS to JBoss AS
  (2) Securing Core Web Services on JBoss AS
  (3) Securing BPEL Service on Oracle BPEL Process Manager

(4) OUTLOOK

Information 2: WUSKAR CASE STUDY -
Content Overview

Diese WUSKAR-Fallstudie ist entlang des Software-Entwicklungsprozesses, wie er in [C&M-I-AE] vorgestellt wird, strukturiert.

(1) In der Analysephase werden aus einer weitgehend technologieunabhängigen geschäftsorientierten Sicht Anforderungen an die zu implementierende Lösung untersucht. Da hinsichtlich der fachlichen Funktionalität auf bereits vorhandene Arbeiten aufgebaut wird, zeigt die Systemsicht innerhalb der Analysephase einen bereits technologiebetonten *„as is“*-Schnappschuss.

(2) Der Schwerpunkt in der Entwurfsphase liegt auf der Architektur, zeigt also welche Komponenten für eine Lösung benötigt werden, deren Anordnung wie sie zusammenhängen. Aus der Architektur wird bestimmt, welche Komponenten zur Lösung des Problems konfiguriert werden müssen, hier wird bereits ein Ausblick über die verschiedenen Konfigurationsmöglichkeiten gegeben.

(3) In der Implementierungsphase wird der Entwurf umgesetzt. Neben der eigentlichen Absicherung der beteiligten Webservices wird auch die Migration der existierenden Basis-Webservices von AXIS auf den JBoss AS beschrieben. Die Implementierung gibt Aufschluss über die tatsächlich verwendeten Produkte, die Verteilung von Schlüsseln, welche zur Absicherung benötigt werden und schließlich die Konfiguration der involvierten Infrastrukturkomponenten.

(4) In einem Ausblick wird die vorgestellte Lösung kritisch beurteilt und auf ungelöste Teilaspekte aufmerksam gemacht. Unter anderem stießen wir bei der Implementierung auf Interoperabilitätsprobleme.

1 ANALYSE

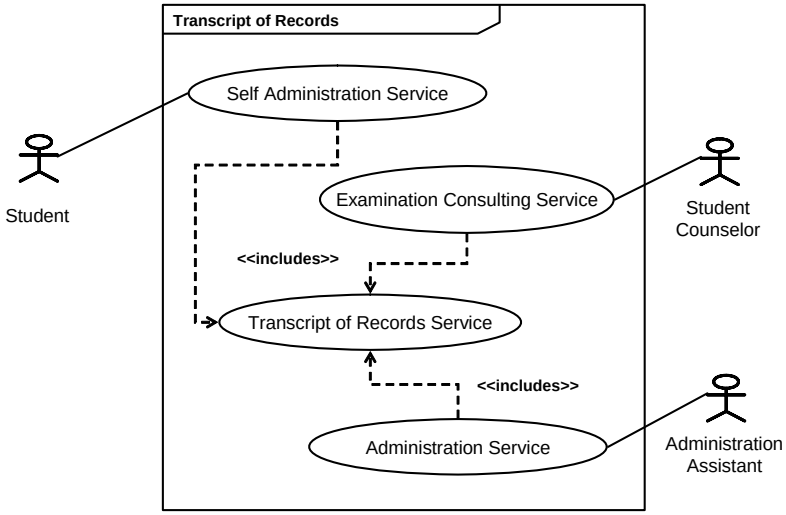
Die Analyse ist in zwei Bereiche aufgeteilt. Im Geschäftsbereich wird der „*Transcript of Records*"-Service aus geschäftlicher Sicht dargestellt sowie Sicherheitsanforderungen an den ToR-Service untersucht. Der Systembereich gibt einen Überblick über die bereits vorhandene allerdings ungesicherte Implementierung des ToR-Services und untersucht Sicherheitsanforderungen aus Systemsicht.

1.1 Geschäftsbereich

Bei einem *Transcript of Records* (ToR) handelt es sich um einen Notenauszug in ECTS-Norm, der einem Studierenden ausweist, welche Leistungen er an der ausstellenden Hochschule erbracht hat. Diesen ToR benötigt er dann wiederum an seiner neuen Hochschule, um sich dort seine bisherigen Studienleistungen anrechnen zu lassen und sein Studium fortsetzen zu können. Die Anrechnung der Studienleistungen erfolgt bei ECTS über den Austausch des Notenauszuges zwischen der Heimat- und der Gasthochschule und umgekehrt, wobei alle beteiligten Parteien (die Heimathochschule, die Gasthochschule und der Studierende) eine unterzeichnete Kopie des ToR erhalten [AE+04].

1.1.1 Anwendungsfälle und beteiligte Rollen

Bei dem „*Transcript of Records*"-Service handelt es sich um einen vollständig automatisierbaren Prozess zur Erstellung eines ECTS-konformen Notenauszuges, der als Service angeboten werden soll.



Information 3: Use Cases including the Transcript of Records Service

Information 3 zeigt drei Anwendungsfälle, die auf diesen „*Transcript of Records*"-Service zurückgreifen.

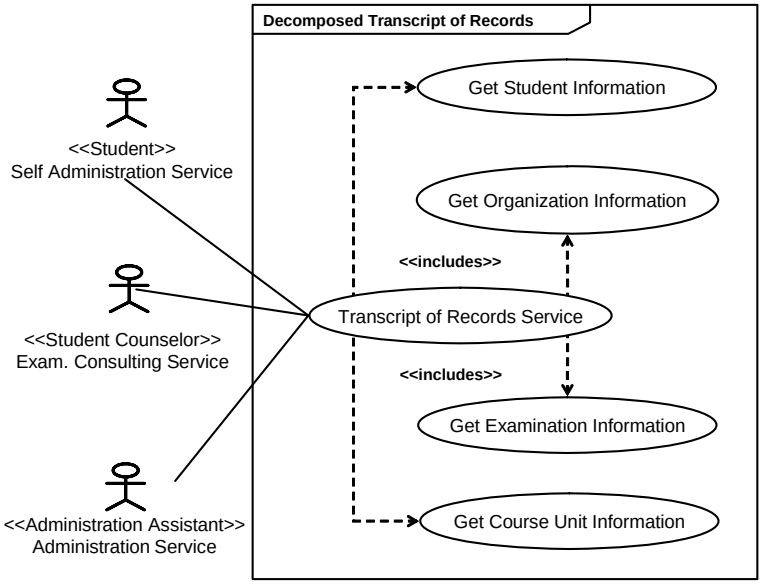
(1) Ein Student soll sich über einen Selbstverwaltungsdienst einen ECTS-genormten Notenauszug selbst ausdrucken können.

(2) Alternativ kann der Notenauszug auch von einem Verwaltungsangestellten der Hochschule (z.B. auf dem Studierenden Sekretariat) erstellt werden.

(3) Ein Studierenden-Berater, z.B. ein Mitglied des Prüfungsausschusses, soll im Rahmen einer Beratung zu Prüfungsfragen ebenfalls einen Notenauszug generieren können.

Ein ToR-Notenauszug besteht im Einzelnen aus den personenbezogenen Informationen eines Studenten, Informationen über abgelegte Prüfungen, Informationen über die Vorlesungen in deren Rahmen Prüfungen abgelegt wurden sowie Informationen zur Fakultät, an welcher der Student eingeschrieben ist.

Aufgrund der (geographischen) Verteiltheit der verschiedenen Informationen, die zur Erstellung eines ToR-Notenauszuges benötigt werden, lässt sich der „*Transcript of Records*"-Service wie folgt verfeinern.



Information 4: Decomposed Use Case: Transcript of Records Service

Die Akteure sind in diesem Fall Systeme, die innerhalb eines konkreten Anwendungsfalls von Vertretern einer bestimmten <<Rolle>> benutzt werden.

(1) Die personenbezogenen Information eines Studenten setzen sich aus privaten (Kontakt- und Adressinformationen) sowie aus hochschulbezogenen Informationen (z.B. Matrikelnummer) zusammen.

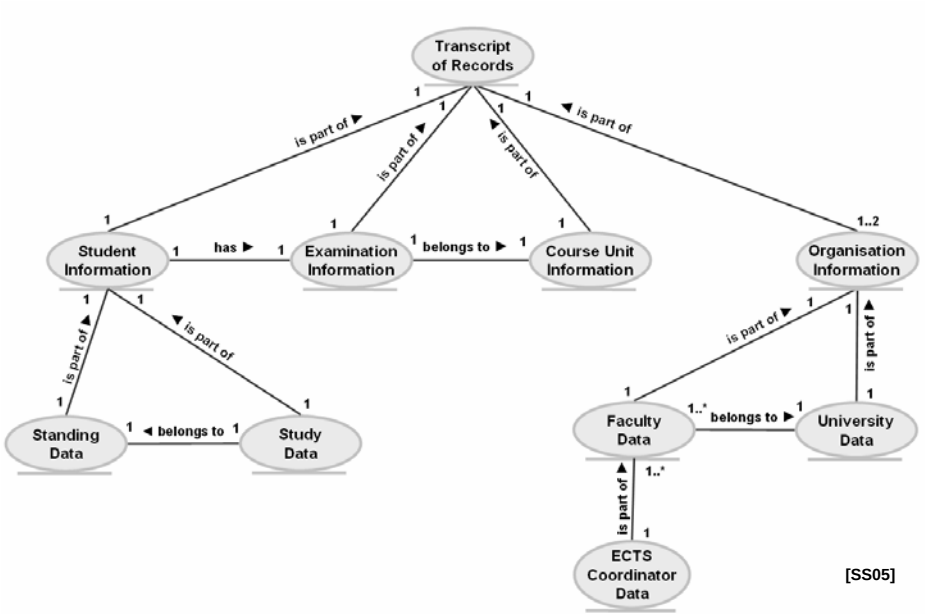
(2) Zu den Fakultätsinformationen gehören unter anderem auch Kontaktinformationen von Ansprechpartnern bzw. Verantwortlichen in ECTS-Fragen.

(3) Die Prüfungsinformationen enthalten alle für den ECTS-Notenauszug relevanten Prüfungsergebnisse des Studenten zusammen mit einer Kennung für die entsprechenden übergeordneten Vorlesungen.

(4) Die Vorlesungsinformationen werden bei der Entscheidung zur etwaigen Anrechnung von ECTS-Punkten in einem ähnlichen Studiengang benötigt.

1.1.2 Geschäftsobjekte

Die von einem ECTS-Notenauszug verwendeten Informationen lassen sich in einem Geschäftsobjektdiagramm folgendermaßen darstellen.

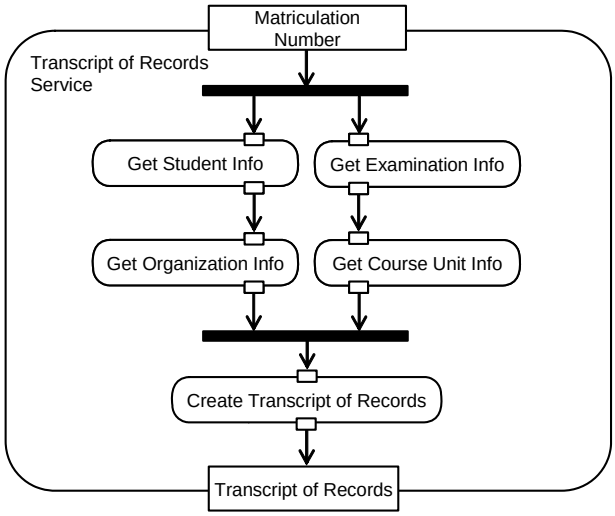


Information 5: Business Object Diagram: Transcript of Records

Hierbei wird deutlich, dass zwischen den verschiedenen Informationen auch gewisse Abhängigkeiten bestehen, z.B. beziehen sich bestimmte Prüfungsinformationen einerseits direkt auf einen Studenten und verweisen andererseits auf den größeren Kontext der zugehörigen Vorlesungsinformationen. Auch Informationen zu einer Fakultät sind in einem Notenauszug nur dann sinnvoll, wenn der Studierende an dieser Fakultät eine Prüfung abgelegt hat.

1.1.3 Geschäftsprozess

Die Abhängigkeiten zwischen den verwendeten Informationen zusammen mit der verfeinerten Darstellung des „*Transcript of Records*"-Services spiegeln sich im Aktivitätsdiagramm (Information 6) wider.



Information 6: Activity Diagram of Use Cases: Transcript of Records Service

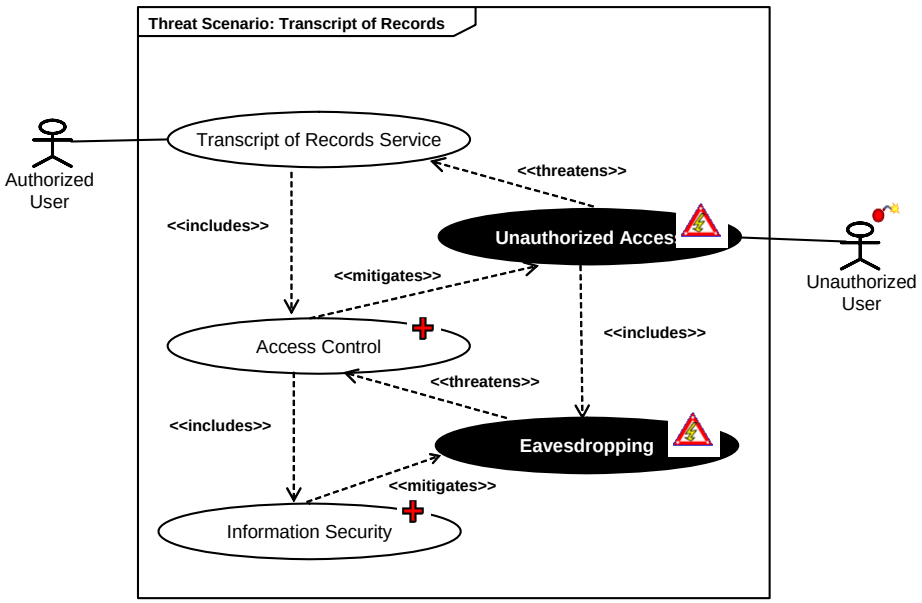
Kästen mit kantigen Ecken stellen Datenobjekte dar (Ein- und Ausgabeparameter einer Aktivität, *data objects*), Kästen mit runden Ecken Aktivitäten bzw. Anwendungsfälle (*working nodes*). Balken stellen Synchronisationspunkte nebenläufiger Aktivitäten dar (*routing nodes*).

(1) Information 6 zeigt den Aktivitäts- und Objektfluss, der dem „*Transcript of Records*“-Service zugrunde liegt. Als Eingabe wird die Matrikelnummer eines Studenten erwartet. Um den Notenauszug zu erstellen, werden zunächst alle benötigten Informationen gesammelt. Hieraus wird schließlich ein ECTS-konformer Notenauszug generiert, der als Ergebnis zurückgegeben wird.

(2) Die hierfür erforderlichen Informationen werden in vier Teilschritten zusammengetragen. Zur Bereitstellung der relevanten Fakultätsinformationen, ist es zunächst erforderlich, die Studierendeninformationen auszuwerten, die Bereitstellung der relevanten Kursinformationen erfordert die vorausgegangene Auswertung der Prüfungsinformationen. Für die Beschaffung der Studenteninformation und der Prüfungsinformationen wird die Matrikelnummer benötigt.

1.1.4 Sicherheitsanforderungen aus Geschäftssicht

Aus Geschäftssicht ergeben sich neben den vorgestellten fachfunktionalen Anforderungen an den „*Transcript of Records*“-Service auch nichtfunktionale Anforderungen, allen voran der Schutz sensibler Daten. Ein anonymisierter Notenspiegel wäre sicher unkritisch, würde allerdings den Vorgaben der ECTS-Norm nicht entsprechen. Schon das Veröffentlichen der Verknüpfung von Name und Matrikelnummer, oder die Bereitstellung von Adressinformationen in einem bestimmten Kontext (z.B. als Studierender an einer Universität) wird im deutschen Recht als Beschneidung der Privatsphäre gedeutet, das Hinzunehmen von Prüfungsergebnissen verschärft diesen Sachverhalt weiter. Daher ist die Vorraussetzung für die Realisierung des „*Transcript of Records*“-Services ein vertraulicher Umgang mit den zugrunde liegenden Informationen.



Information 7: Threat Scenario: Transcript of Records Service

Da der ECTS-Notenauszug auch personenbezogene Information enthält und mit Webservice-Technologien zur Verfügung gestellt werden soll, müssen die Information geschützt und der Zugriff auf diese kontrolliert werden.

(1) Der Betrieb des „*Transcript of Records*“-Services ist gefährdet, solange unbefugten (feindlichen) Nutzern unkontrollierter Zugang gewährt wird.

(2) Durch Einrichten einer Zugriffskontrolle wird diese Bedrohung gemildert. Aber auch die Wirksamkeit der Zugriffskontrolle ist fraglich, solange Informationen im Transit abgefangen werden können.

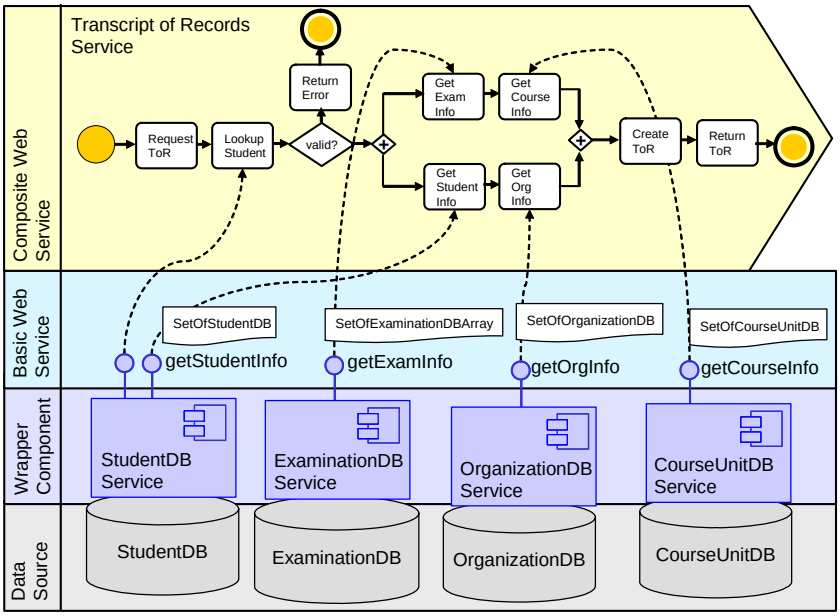
(3) Durch die Absicherung der Information wird das einfache passive Mithören verhindert.

Die Bedrohungen sowie die Gegenmaßnahmen vererben sich direkt auf die sekundären Aktivitäten des „*Transcript of Records*"-Services und den von diesen bereitgestellten Informationen.

Vom „*Transcript of Records*"-Service selbst geht nur ein sehr geringes Sicherheitsrisiko aus, da Informationen nicht verändert sondern nur gelesen werden. Selbst ein verfälschter oder gänzlich unbrauchbarer Notenauszug führt zu keinem maßgeblichen Schaden. Die verschiedenen Nachrichten, die während der Erzeugung eines ToR ausgetauscht werden, sollten dennoch gegen unautorisiertes Abändern geschützt werden.

1.2 Systembereich

Im Rahmen mehrer WUSKAR-Fallstudien (siehe Kapitel 0.2) wurde der „*Transcript of Records*"-Prozess bereits automatisiert und unter Verwendung von Webservice-Technologien implementiert. Zunächst wurden zu diesem Zweck bereits vorhandene Datenquellen mit Webservice-Schnittstellen versehen, das bedeutet, dass ein Teil der Funktionalität über eine WSDL-Beschreibung exponiert wird und mittels SOAP abgerufen werden kann.



Information 8: ToR BPEL Process interacting with Web Service Components

(1) Der eigentliche ToR-Service wurde als BPEL-Prozess implementiert, Information 8 deutet diesen durch eine vereinfachte BPMN-Notation an. Zusätzlich zu den in Information 6 dargestellten Aktivitäten wurde ein „Plausibilitätstest" eingefügt, der vor Ausführung des Prozesses die übergebene Matrikelnummer auf Gültigkeit prüft. Der BPEL Prozess wird über SOAP angesprochen und erledigt seinerseits Aufrufe nach außen über SOAP. Die ausführende Einheit ist eine BPEL-fähige Prozess-Engine.

(2) Die Webservice-Schicht stellt Ausschnitte aus den WSDL-Beschreibungen der implementierten Basis-Webservices dar, nämlich Operationsnamen und zugehörige komplexe

Datentypen. Darüber hinaus wird angedeutet innerhalb welchem Prozessaktivitätsschritt eine bestimmte Operation aufgerufen wird. Die Kommunikation zwischen Service und Client geschieht über SOAP. Die ausführende Einheit ist eine SOAP-Engine.

(3) Die Komponentenschicht zeigt die *Wrapper*-Komponenten, welche einen Teil der Funktionalität der Datenquellen über Webservice-Schnittstellen exponiert. Verwendet werden hier Java-Komponenten, die Abbildung zwischen Java-Klasse und Webservice-Schnittstelle geschieht durch die Java-XML-APIs JAX-RPC bzw. JAX-WS. Die Ausführende Einheit ist ein Komponenten-Container.

(4) Aus Sicht der Webservice-basierten SOA sind die vorhandenen Datenquellen Altsysteme (*legacy system*), da sie nicht von Haus aus mit den entsprechenden Schnittstellen (WSDL, SOAP) ausgestattet sind. Angesprochen werden die Datenquellen aus Komponentensicht via SAP JavaConnector (JCo) bzw. Java Database Connectivity (JDBC).

1.2.1 Vorhandene Systeme

Die bereits vorhandene rein fachfunktionale Lösung wird auf einer Infrastruktur betrieben, die aus folgender Software aufgebaut ist.

(1) BPEL process engine
  (1) Oracle Application Server
  (2) Oracle BPEL Process Manager

(2) Web Service enabled component container
  (1) Apache Jakarta Tomcat web container
  (2) Apache AXIS SOAP engine

(3) Data sources
  (1) SAP Campus Management
  (2) MySQL databases

Information 9: Existing Systems

(1) Zum Betrieb des BPEL-Prozesses wurde der Oracle BPEL Process Manager verwendet.

(2) Zum Betrieb der Basis-Webservices wurde der Web-Container Apache Jakarta Tomcat zusammen mit der SOAP-Engine Apache AXIS verwendet.

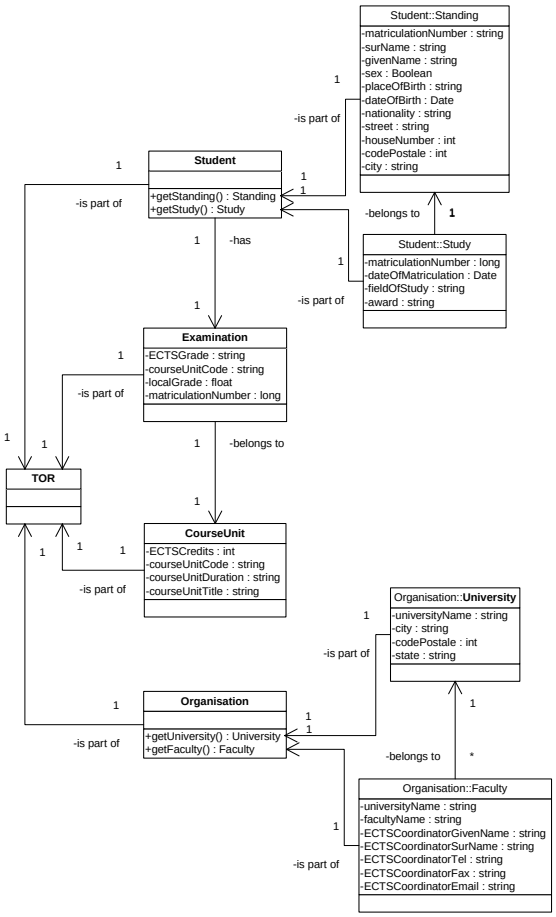
(3) Bei den zugrunde liegenden Datenquellen handelt es sich um MySQL-Datenbanken und ein SAP Campus Management System.

SAP Campus Management

Das Software-System SAP R/3 wurde für den Einsatz in Unternehmen und Institutionen entwickelt mit dem Ziel einer computergestützten Verwaltung von Daten. Es besteht aus Modulen, die verschiedene Bereiche umfassen (z.B. Vertrieb, Finanzwesen, Personalwesen). SAP Campus Management ist ein Modul, das in das SAP R/3-System eingehängt werden kann. Es enthält Bildungsmanagementfunktionalität für Hochschulen und ist in folgenden Teilkomponenten unterteilt.

1.2.2 Darstellung der Geschäftsobjekte

Ein ToR-Objekt besteht aus allen relevanten Informationen eines Studenten, wenn der Hochschulen wechseln möchte.



Information 10: Business Object Model: Transcript of Records

Diese Informationen werden in vier Bereiche aufgeteilt, nämlich Studenten-, Klausur-, Vorlesungs- und Organisationsinformation. Das Ziel der Erstellung eines ToR-Objektes ist es, Prüfungen in Verbindung mit dem europäischen Leistungspunktessystemen (ECTS) einheitlich anzurechnen.

(1) Studenteninformationen bestehen aus Personaldaten wie z.B. Vorname und Familienname, Nationalität, Adresse oder Herkunft und Studieninformationen z.B. wie Matrikelnummer, Einschreibungsdatum, Fachrichtung oder Stipendium.

(2) Klausurinformationen beinhalten die Ergebnisse aller Klausuren und Prüfungen, die ein Student erfolgreich absolviert hat. Klausurinformationen stehen im Zusammenhang mit bestimmten Vorlesungsinformation, da eine einzelne Vorlesung oder ein Veranstaltungsblock eine entsprechende schriftliche oder mündliche Prüfung fordert.

(3) Vorlesungsinformationen enthalten eine eindeutige Vorlesungskennung, einen Titel, die Dauer und die Anzahl der anrechenbaren ECTS-Leistungspunkte.

(4) Die Organisationsinformationen beziehen sich auf Fakultäten und Hochschulen, die von einem Studenten besucht wurden.

1.2.3 Sicherheitskonzepte und Sicherheitstechnologien

Bevor das SOAP-Protokoll entwickelt wurde, wurden Nachrichten oft mit Hilfe komplizierter Protokolle und unverständlichen, weil proprietären Inhalten ausgetauscht. Verschlüsselung und Signierung wurden nicht in heutigem Ausmaß berücksichtigt, da Nachrichten schwer zu

dechiffrieren oder zu verändern waren und ein entsprechender Zusatzaufwand bei niedrigen Datenübertragungsraten kaum gerechtfertigt werden konnte. SOAP-Nachrichten sind im Allgemeinen deklarative XML-Dokumente, die in einem gewöhnlichen Texteditor gelesen und modifiziert werden können. Zudem sollen SOAP-Nachrichten vor allem zur Nutzung über das Internet eingesetzt werden und sind daher besonders schutzbedürftig. Beim geschäftlichen Einsatz entstehen zusätzliche Sicherheitsbedürfnisse.

(1) Authentication

(2) Authorization

(3) Confidentiality

(4) Integrity

Information 11: Security Requirements

(1) Bei der Authentifizierung handelt es sich um den Nachweis der Identitätsbehauptung eines Gegenübers innerhalb einer Kommunikationsbeziehung. Die Authentifizierung des Gegenübers wird in Netzwerken häufig eingesetzt, um die Zugriffsberechtigung zu überprüfen. Die reine Identifizierung beschreibt ein Verfahren, durch das ein Benutzer oder ein Rechner anhand von üblicherweise eindeutigen Merkmalen über ein Netzwerk erkannt werden kann. Ein Benutzername beispielsweise ermöglicht die Identifizierung einer Person oder eine IP-Adresse ermöglicht die Identifizierung eines Rechners. Bei diesem Verfahren wird keine Überprüfung der Echtheit der Identifikationsbehauptung eines Teilnehmers betätigt. Mit einfachen Mitteln kann ein Teilnehmer seine Identität verfälschen oder die Identität eines Anderen vortäuschen. Im Gegensatz hierzu findet bei einer Authentifizierung eine Überprüfung der Behauptung statt, ob ein Benutzer oder ein Rechner auch tatsächlich derjenige ist, für den er sich ausgibt. Eine sichere Authentifizierung wird gewährleistet durch Kombination unterschiedlicher Merkmale. Eine Kombination von mindestens zwei der folgenden drei Merkmale wird als „starke Authentifizierung“ bezeichnet:

- Etwas wissen (*something you know*), z.B. ein Passwort.
- Etwas haben (*something you have*), z.B. eine Smartkarte.
- Etwas sein (*something you are*), z.B. Fingerabdruck.

(2) Autorisierung ist der Vorgang, bei welchem einzelnen Subjekten, Rollen und Gruppen Zugriffsrechte auf Ressourcen wie Informationen, Daten und Dienste innerhalb eines Informationssystems übergeben werden. Autorisierung ist damit die Grundlage für die Verwaltung und Überprüfung von Zugriffsrechten. Für den Nutzbetrieb ist vor allem die Zugriffskontrolle von Bedeutung. Nach der Ermächtigung kann die gewünschte Ressource freigegeben oder die gewünschte Transaktion ausgeführt werden. Normalerweise erfolgt eine Zugriffskontrolle nach einer erfolgreichen Authentifizierung.

(3) In einem Informationssystem bezeichnet Vertraulichkeit die Eigenschaft von Informationen nicht für Unbefugte zugänglich zu sein.. Beispielsweise können die ausgetauschten Nachrichten durch Verschlüsselung und Entschlüsselung nur für Sender und Empfänger als Klartext lesbar sein. Ein unbefugter Dritter müsste alle möglichen Schlüssel durchprobieren, um an den Inhalt der Nachrichten zu gelangen.

(4) Die Integrität ist die Eigenschaft von Informationen vor nachträglicher Veränderung geschützt zu sein, d.h. man kann feststellen, ob die Daten beim Transport manipuliert wurden oder verhindern, dass sie manipuliert werden können. Allerdings ist die Integrität der Daten mit Hashing-Algorithmen nicht unbedingt zuverlässig gewährleistet, da ein unbefugter Dritter die Daten einfach modifizieren und dann den passenden neuen Hash-Wert an die Daten anhängen

könnte. Um dies zu verhindern, werden digitale Signaturen eingesetzt. In diesem Bereich wird eine besondere Eigenschaft des asymmetrischen RSA-Algorithmus genutzt. Eine digitale Signatur wird dadurch erzeugt, indem zunächst ein Hash-Wert der zu sendenden Daten berechnet wird und diesen anschließend mit einem privaten Schlüssel verschlüsselt. Danach fügt man diese digitale Signatur an die Daten an. Diese Daten können von jedermann mit dem passenden öffentlichen Schlüssel geprüft werden.

(1) Secure Socket Layer

(2) XML Encryption

(3) XML-Signature

(4) WS-Security

Information 12: Security Technologies

(1) *Secure Socket Layer* (SSL) bezeichnet ein Verschlüsselungsverfahren für Daten zwischen Webbrowser und Webserver über das HTTP-Protokoll. Dabei ist SSL in der Lage, eine sichere Datenübertragung zwischen zwei direkt miteinander kommunizierenden Partnern auf der Transportebene zu garantieren. Wenn es aber erforderlich ist, einen Sicherheitskontext herzustellen, der über mehrere Partner hinweg besteht, stößt SSL an seine Grenzen.
Bei der Webservice Kommunikation handelt es sich oft um eine „Ende-zu-Ende“-Verbindung, das bedeutet, zwischen dem Service, der eine Anfrage entgegennimmt und dem Service, der die Anfrage bearbeitet, können mehrere Services liegen, die die Anfrage weiterleiten. Die Verwendung von SSL begrenzt die Absicherung der Webservices auf „Punkt-zu-Punkt“-Verbindungen auf der Übertragungsroute zwischen den Services. Da Webservices für die Kommunikation nur einseitig sichtbar sind, reicht SSL für Authentifizierung und Autorisierung der SOAP-Anfrage auf der Sessionschicht nicht aus. Die SSL-Authentifizierung kann zur Identifizierung der Endbenutzer nicht verwendet werden, denn der Webservice kann nicht feststellen, von wem die SOAP-Anfrage generiert wurde.

(2) Das Ziel von XML Encryption ist es, die vollständige, beziehungsweise Teile einer SOAP-Nachricht geheim zu halten. XML Encryption ist ein XML-Security-Standard des W3C. Durch XML Encryption ist es möglich, gesamte XML-Dokumente beziehungsweise Teile daraus mit einem oder evtl. mehreren verschiedenen Schlüsseln zu verschlüsseln. Die Verwendung mehrerer verschiedener Schlüssel ist für unterschiedliche Empfänger einer Nachricht geeignet, wobei jeder der Empfänger nur einen bestimmten Teil dieser Nachricht entschlüsseln darf. Und die anderen nicht für diesen Empfänger freigegebenen Teile der Nachricht sollen verborgen bleiben.

(3) XML-Signature ist ebenfalls ein W3C XML-Security-Standard, mit dem in erster Linie die Integrität des Nachrichtenaustausches gewährleistet werden soll. XML-Signaturen bedienen sich dazu dem Prinzip digitaler Signaturen. Die XML-Nachricht wird nicht verändert, sondern durch die Signatur ergänzt. Eine XML-Nachricht kann auf Veränderungen mithilfe dieser digitalen Signatur überprüft werden. XML-Signaturen müssen für alle Ressourcen unter einem gleichen URI inklusiv die nicht in XML-Format Ressourcen wie Bilder oder HTML-Dateien angewendet werden. Außerdem müssen XML-Signaturen für Teile eines Dokumentes oder das gesamte Dokument verfügbar sein. Wenn ein XML-Dokument von mehreren Benutzern bearbeitet werden soll, könnte der bestimmte Teil dieses XML-Dokumentes für den berechtigten Benutzer entsprechend unterschiedlicher Signaturen freigestellt werden. Die Teile ohne Signaturen können für weitere Bearbeitung verwendet werden. XML-Signaturen müssen beliebige kryptographische Signaturmethoden unterstützen.

(4) Web Services Security (WS-Security oder WSS) bietet Maßnahmen zur Absicherung der SOAP-Nachrichten durch Integrität, Vertraulichkeit der Nachrichten und Authentifizierung für die einzelnen Nachricht an. Eine ganze Reihe von Sicherheitsmodellen und Verschlüsselungstechnologien können als WS-Security-Maßnahmen verwendet werden. WS-Security Spezifikationen beschreiben Mechanismen, wie man SOAP-Nachrichten über Webservices sicher austauschen kann. Diese WS-Security-Basismechanismen können miteinander kombiniert werden.

1.2.4 Ausblick: WS-Security und verwandte Standards

Seit 2001 wurden WS-Security Spezifikationen entwickelt und im April 2002 von IBM und Microsoft veröffentlicht. OASIS hat einige Teile im Juni 2002 zur Standardisierung übernommen. Andere Unternehmen und Organisationen wie BEA, HP, Novell, SAP, SUN und Computer Associates haben dazu auch beigetragen.

Das Hauptziel der WS-Security ist, normale SOAP-Nachrichten mit Sichheits-Elementen (Sicherheitstoken) in Verbindung zu bringen. Für WS-Security sind keine spezifischen Typen von Sicherheits-Elementen erforderlich. Dabei können die bestehenden Absicherungstechnologien für Webservices wie XML Encryption und XML-Siganture integriert werden. Diese Technologien sind für die WS-Security modular gehalten. WS-Security stellt fest, wie Sicherheitsmerkmale eingebunden werden. Die Merkmale und Verfahren zur Absicherung kann man frei auswählen. Deswegen erfasst WS-Security Framework alle Spezifikationen für die Integration in bereits bestehende Umgebungen.

(1) WS-Security: SOAP Message Security

(2) WS-Policy Framework

(3) WS-Trust Language

(4) WS-SecureConversation

(5) WS-Federation

(6) WS-Privacy

(7) WS-Authorization

Information 13: Outlook: WS-Security and Related Specifications

(1) *SOAP Message Security* beschreibt eine Erweiterung des SOAP-Protokolls zur Gewährleistung von Vertraulichkeit und Integrität beim Austausch einzelner SOAP-Nachrichten. SOAP Message Security bedeutet, wie Sicherheitstokens in SOAP-Nachrichten eingebunden und codiert werden. Sicherheitstokens bestehen aus beispielsweise persönlichen Nutzer-Informationen, Schlüssel oder sogar Zertifikaten. Eine reguläre SOAP-Nachricht kann zu einer abgesicherten SOAP-Nachricht durch einen Sicherheits-Header hinzugefügt werden. Der Sicherheits-Header enthält Sicherheitstoken und digitale Signaturen. Sicherheitstoken werden von Username Token Profile 1.0 und X.509 Certificate Token Profile 1.0 Spezifikationen definiert. Digitale Signaturen werden nach der XML-Sigature Spezifikation erstellt.

(2) *Web Services Policy* (WS-Policy) beschreibt Anforderungen, Präferenzen, vorhandene Fähigkeiten und Zusicherungen eines Webservices innerhalb einer SOAP-Nachricht. Beispielsweise kann eine Policy anzeigen, dass ein Webservice nur eingehende Anfragen akzeptiert, wenn diese Policy eine gültige Signatur enthält oder die Nachricht eine bestimmte Größe nicht überschreitet. Damit kann ein Webservice eindeutig mitteilen, welche Authentifizierungsverfahren er unterstützen kann und welches er bevorzugt oder welche Verschlüsselungsart er bei der Übertragung mindestens erwartet. Außerdem bezeichnet WS-

Policy Spezifikation, wie eine Policy verteilt bzw. bezogen werden kann. *Web Services Policy Attachment* (WS-Policy Attachment) und *Web Services Metadata Exchange* (WS-Metadata Exchange) Spezifikationen definieren, wie eine Policy über SOAP-Nachrichten oder eingeschlossene XML und WSDL-Dokumenten zugänglich gemacht werden kann.

(3) *Web Services Trust* (WS-Trust) baut auf dem Framework von WS-Security auf und legt SOAP-basierte Mechanismen für das Vermitteln von Vertrauensbeziehungen sowie für das Anfordern und Zurückgeben von Sicherheitstokens fest. Ein Sicherheitstoken-Dienst kann angefordert werden, wenn ein Sicherheitstoken mit bestimmten Eigenschaften zurückgegeben werden muss. Diese Anforderung basiert auf einem bereits vorhandenen Sicherheitstoken, das dem Aufrufer und dem Sichheitstokendienst nicht völlig unbekannt ist. WS-Trust erweitert WS-Security durch Ausstellung, Erneuerung und Bestätigung von Sicherheitstoken zwischen Webservices und einem Sicherheitstokendienst. Dabei werden sichere Verbindungen etabliert, beurteilt und vermittelt.

(4) *Web Services Secure Conversation* (WS-SecureConversation) bietet auf SOAP Message Security und WS-Trust basierend Mechanismen an, um eine sichere Kommunikation beim Austausch mehrer Nachrichten zu realisieren. Das Ziel des WS-Secure-Conversations ist, ein Sicherheitskontext zu teilen und Sicherheitsschlüssel aus einem Sicherheitskontext abzuleiten.

(5) *Web Services Federation* (WS-Federation) definiert einen auf WS-Plicy, WS-Trust und WS-Security basierten Mechanismus, wie Zusammenschlüsse mehrerer Vertrauenszonen in einem Verbund gebraucht werden.

(6) *Web Services Privacy* (WS-Privacy) definiert keine neuen Policy-Sprachen, sondern das Zusammenspiel zwischen WS-Trust, WS-Policy und WS-Security, um automatische Bekanntgabe bzw. Einhaltung von Datenschutzrichtlinien innerhalb einer SOAP-Kommunikation zu ermöglichen. Diese Spezifikation bleibt bis heute noch weiter in der Aufbauphase.

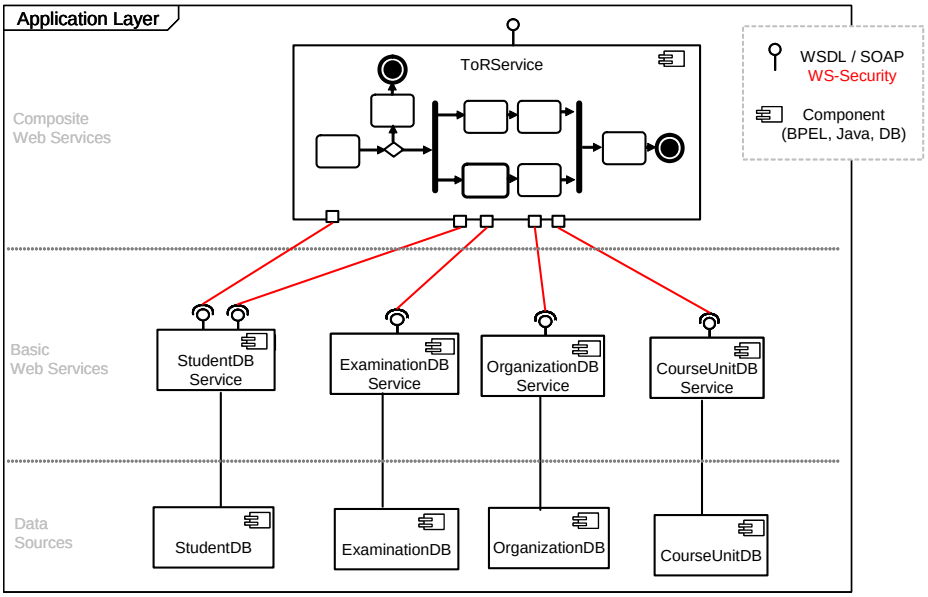
(7) *Web Services Authorization* (WS-Authorization) beschreibt, wie Autorisierungsdaten und –policies in einem Webservice spezifiziert und verwaltet werden. WS-Authorization gewährt Flexibilität und Erweiterbarkeit ohne das Autorisierungsformat oder die Autorisierungssprache zu verletzen. Ein Webservice kann den Autorisierungsdienst mit einer Menge von Claims und Sicherheitstoken aufrufen. Dabei sucht der Autorisierungsdienst die gültige Authorization Policy und ermittelt, welche Zugriffsrechte vergeben werden müssen. Dannach sendet der Autorisierungsdienst das Ergebnis der Autorisierung zurück, entweder durch Herausgeben eines Authorization-Tokens oder in dem er mit einigen Nachrichten antwortet. Allerdings ist diese Spezifikation noch nicht veröffentlicht.

1.2.5 Sicherheitsanforderungen aus Systemsicht

Jedes Mal beim Aufruf eines ToR-Services aus anderen Geschäftsprozessen heraus werden die folgenden Schritte durchgeführt. Am Anfang wird eine Anfrage für ein ToR-Objekt an den ToR-Service geschickt, wobei die Matrikelnummer als Parameter dieser Anfrage gesendet wird. Wenn ein Student mit dieser Matrikelnummer ordentlich eingeschrieben ist, werden die Basis-Webservices ausgeführt, um alle nötigen Informationen für diesen Student zurückzuliefern. Ansonsten wird eine Fehlermeldung als Antwort erzeugt. Nur wenn alle Informationen gültig sind, kann ein ToR-Objekt für den aufrufenden Geschäftsprozess erstellt werden.

Bei dieser Ausführung spielt die Authentifizierung und Überprüfung der Zugriffsberechtigung des Nutzers des ToR-Services eine große Rolle. Die weiteren Funktionalitäten der Basis-Webservices hängen ebenfalls von dieser Authentifizierung des Nutzers ab. Um eine sichere Authentifizierung zu ermöglichen müssen Benutzerinformationen, wie z.B die Matrikelnummer

in der Anfrage verschlüsselt werden. Anderseits müssen auch die von den Basis-Webservices zurück gelieferten Informationen unter die Berücksichtigung durch Webservice-Sicherheit gebracht werden, um die privaten Daten gegen unerlaubtes Abfangen zu schützen.



Information 14: Securing exchanged Messages within the Transcript of Records Service

Die Aufgabenstellung der WUSKAR-Fallstudie besteht nun darin, den Nachrichtenaustausch zwischen den Services abzusichern und dabei die Interoperabilität zu bewahren. In Frage kommt hierzu der OASIS-Standard WS-Security. WS-Security stellt Mechanismen zur Wahrung von Vertraulichkeit und zum Nachweis der Unversehrtheit von Informationen auf Nachrichtenebene und zum Mitsenden von *Security Tokens* innerhalb einer Nachricht bereit. Die Absicherungsmechanismen beziehen sich dabei auf einzelne Nachrichten bzw. auf Teile dieser Nachrichten und werden auch über Zwischenknoten (SOAP *intermediaries*) hinweg aufrechterhalten. Dadurch ist eine „Ende-zu-Ende“-Absicherung von kritischen Inhalten möglich. Zudem impliziert die direkte Absicherung der kritischen Informationen Unabhängigkeit vom Transport-Protokoll. Eine abgesicherte Nachricht könnte sogar persistent verwahrt werden, damit bleiben Nachrichten auch bei einer Zwischenspeicherung geschützt. Bei der Authentifizierung sollten zwei Gesichtspunkte unterschieden werden: Die Authentifizierung der Nachricht, also der Nachweis der Herkunft (bzw. Echtheit) der Nachricht und die Authentifizierung eines Nutzers (bzw. Senders).

Zur Verbesserung der Portierbarkeit der *Wrapper*-Komponenten, werden die Basis-Webservices zunächst von der bestehenden Tomcat/AXIS-Konfiguration auf den JBoss Application Server umgesiedelt. Dieser ist vollständig J2EE-konform implementiert und richtet sich vor allem nach den Java-Standards des J2EE 1.4-Frameworks und teilweise bereits auch nach einigen Empfehlungen aus Java EE 5. Im Gegensatz hierzu handelt es sich bei den *Open Source*-Implementierungen der Apache-Körperschaft um eigenständige Lösungen. Webservices müssen speziell für AXIS entwickelt werden und können dann nur mit Hilfe von AXIS betrieben werden (Siehe hierzu auch 3.1).

1.3 Übungsaufgaben

Hinweise zur Lösung der Übungsaufgaben befinden sich am Ende des Dokumentes im Kapitel Lösungshinweise zu den Aufgaben.

?

(1) To create a Transcript of Records some distributed informations are to be collected. Figure out the required informations!

(2) Compare Information 6 with the process in Information 8, where are the differences? Try to notate the process with BPEL's basic and structured activities!

(3) Within this scenario, where are the benefits of using web service technologies? Is there any alternative?

(4) The ToR Service deals with sensible data. Describe security requirements from a business (and from a systems) point of view!

Information 15: ANALYSIS - Exercises

(1) Zur Erstellung eines „*Transcript of Records*"-Notenauszuges werden zunächst verteilte Information gesammelt. Um welche Informationen handelt es sich hierbei?

(2) Vergleiche Information 6 mit dem Prozess in Information 8, wo bestehen Unterschiede? Notiere den Prozess in der Business Process Execution Language (BPEL) unter Verwendung von *basic activities* und *structured activites*!

(3) Welchen Vorteil bringt die Verwendung von Webservice-Technologien in diesem Szenario? Wo werden hier Webservice-Technologien eingesetzt, gibt es hierfür Alternativen?

(4) Der „*Transcript of Records*"-Service verarbeitet sensible Informationen. Beschreibe grundlegende Sicherheitsanforderungen aus rein geschäftlicher Sicht (aus technischer Sicht)!

2 ENTWURF

Der nächste Schritt in Richtung Implementierung besteht in der technologie-betonten Verfeinerung des unscharf umrissenen Lösungsansatzes aus der Analyse. Dazu gehört die Untersuchung Einfluss nehmender Faktoren sowie eine zunehmende Konkretisierung in der Modellierung.

2.1 Randbedingungen und Überblick

Nimmt man die involvierten Systeme etwas genauer unter die Lupe, stößt man sehr bald auf Einschränkungen, welche auch die Architektur der Lösung beeinflussen.

(1) Oracle WS-Security restrictions
  (1) Oracle BPEL PM only supports the Username Token
  (2) No support for XML Encryption and XML-Signature
  (3) Oracle Web Services Manager required for WS-Security

(2) JBoss WS-Security restrictions
  (1) Username Token is not fully supported

(3) Restrictions of both implementations
  (1) Encryption / signature processing only with X.509 certificates
  (2) Static keys to encrypt response messages

Information 16: System Restrictions

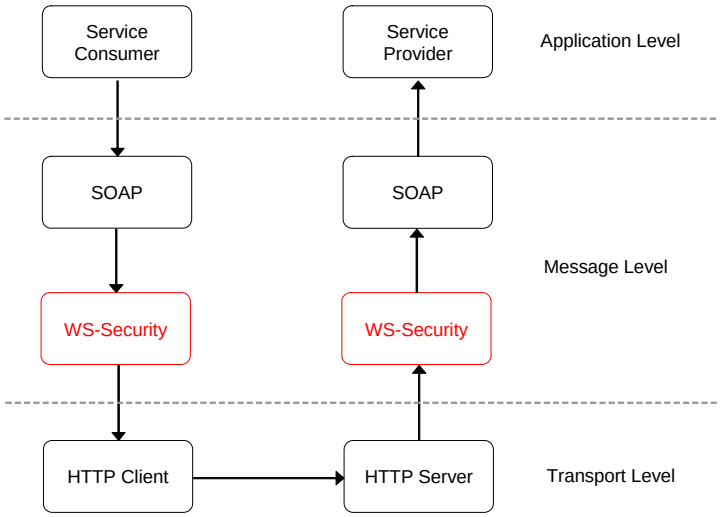
(1) Der Oracle BPEL Process Manager beispielsweise spricht von Haus aus kein WS-Security. Auch durch die Integration des Process Managers in den Oracle AS erhält man wenig mehr WS-Security-Funktionalität für ein- und ausgehende SOAP-Nachrichten eines BPEL-Prozesses. Eine SOAP-Nachricht kann lediglich mit einem Security-Header <wsse:Security> versehen und in diesen ein *Username Token* gelegt werden. Unterstützung für XML-Signature und XML Encryption sowie das X.509-Tokenformat liefert erst eine zusätzliche Komponente, der Oracle Web Services Manager. Als eigenständiger Prozess fungiert er als *Security Gateway* und befindet sich damit außerhalb des BPEL-Containers.

(2) Auch der JBoss Application Server weißt in der verwendeten Version einige Beschränkungen bei der Umsetzung von WS-Security auf, die für den Entwurf beachtet werden müssen. Das *Username Token* wird nur unzureichend (teilweise proprietär) unterstützt.

(3) Zur Berechnung von Signaturen und Verschlüsselungen lassen sich ausschließlich Schlüssel aus X.509-Zertifikaten verwenden. Diese müssen zudem noch „verbindlich“ in verschiedenen *Keystores* abgelegt werden. Es muss also bereits während der Entwurfsphase bestimmt werden, welcher Schlüssel für die Verschlüsselung / Signierung ausgehender Nachrichten eingesetzt werden soll (gilt auch für OWSM).

2.1.1 Signieren und Verschlüsseln mit WS-Security

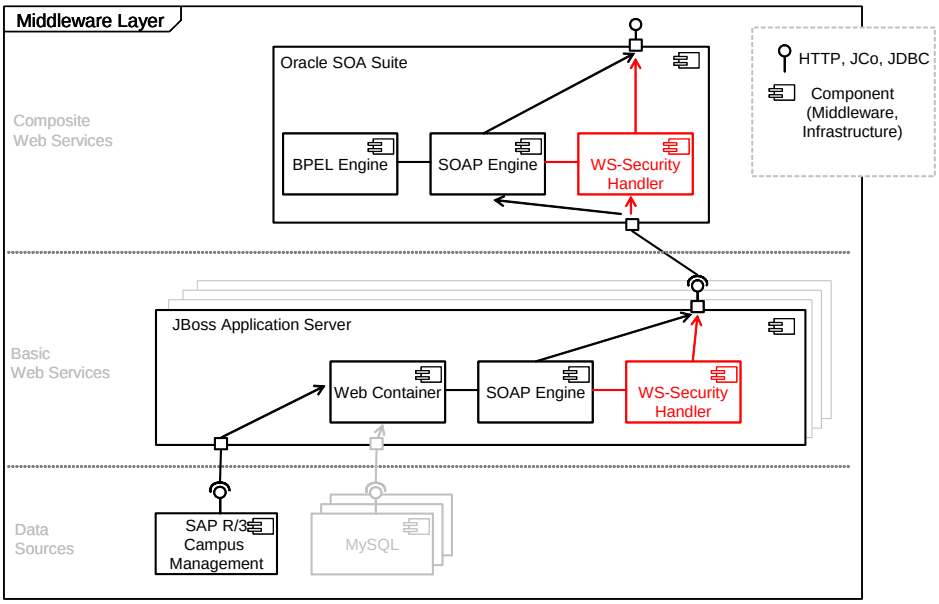
Zur Wahrung der Vertraulichkeit und zur Verhinderung einer unbemerkten Abänderung von SOAP-Nachrichten sollen der Nachrichtensicherheitsstandard WS-Security eingesetzt werden.



Information 17: Web Service Interaction Levels

(1) Bei WS-Security handelt es sich um eine Protokollbeschreibungssprache, welche das SOAP-Protokoll um sicherheitsrelevante Elemente erweitert. Die Einordnung in den SOAP-Stack ist etwas diffiziel, WS-Security sieht nicht nur die Elemente einer SOAP-Nachricht (Body, Header), was für die Einordnung analog zu Information 17 sprechen würde, sondern kann gezielt auf konkrete Nachrichteninhalte angewandt werden, was für eine Einordnung oberhalb von SOAP sprechen würde. WS-Security im SOAP-Stack als SOAP-Erweiterung zu notieren würde wiederum den meisten Implementierungen widersprechen. Die Pfeilrichtung gibt die Reihenfolge der Aufrufe an.

(2) Bei der Betrachtung von Information 17 wäre alternativ auch eine Absicherung der Kommunikation auf Transportebene denkbar, allerdings würde dabei die Flexibilität des SOAP-Protokolls auf das Niveau von HTTP abgesenkt. Um die Möglichkeit für eine „Ende-zu-Ende“-Absicherung über SOAP-Zwischenknoten hinweg offen zu lassen, kommt hier WS-Security zum Einsatz. Zudem wird hierdurch vom Transportprotokoll abstrahiert.



Information 18: Architecture Overview - Middleware Layer

Information 18 gibt einen groben Eindruck, welche Komponenten auf dem *Middleware Layer* in einer Zielarchitektur benötigt werden. Oberhalb des *Middleware Layers* liegt der *Application Layer* (siehe Information 14), darunter wäre ein *Networked Systems Layer*. Die schwarzen Pfeile skizzieren hier den bisherigen Kommunikationspfad vom Aufruf des BPEL-Prozesses bis zur Datenquelle, welcher im Zuge der Absicherung durch die rot markierten Komponenten erweitert wird. Die Pfeilrichtung entspricht hier der Richtung des Angebots bzw. dem Nutzdatenfluss.

(1) Zur Absicherung des Nachrichtenaustausches zwischen Webservices benötigt jede beteiligte Partei eine Komponente, die WS-Security-Protokolle verarbeiten kann. Bei WS-Security handelt es sich zwar um eine SOAP-Erweiterung, eine Erweiterung der SOAP-Engine selbst wäre hier durchaus möglich, trotzdem werden in der Praxis ausgelagerte Komponenten wie *Interceptors* und *Gateways* bevorzugt. Ein *Interceptor*, oder *Agent*, ist eine Komponente innerhalb desselben Serverprozesses, welche ein- und ausgehende SOAP-Nachrichten abfängt und weiterverarbeitet. Ein *Gateway* dagegen ist die serverseitige Entsprechung zum Proxy, also ein eigenständiger Serverprozess. Aus SOAP-Sicht ist ein *Gateway* ein intelligenter Zwischenknoten.

(2) Zur Nutzung von WS-Security muss die entsprechende fachfunktionale Komponente beim jeweils vorhanden WS-Security-Handler registriert werden.

(3) Weiter muss dem WS-Security-Handler das konkrete WS-Security-Protokoll, oft in Form einer Konfigurationsdatei, mitgeteilt werden. Es beschreibt wo sich die zu verwendenden Schlüssel oder Token (Zertifikate) befinden und wie diese genutzt werden sollen (Verschlüsselung, Signierung, Authentifizierung).

(4) Zur Verschlüsselung bzw. Signierung von Nachrichteninhalten mit WS-Security werden Schlüssel benötigt, zum Bsp. X.509-Zertifikate. Aufgrund der oben erwähnten Einschränkungen der aktuellen WS-Security-Implementierung auf dem JBoss AS bleibt hier auch keine andere Wahl.

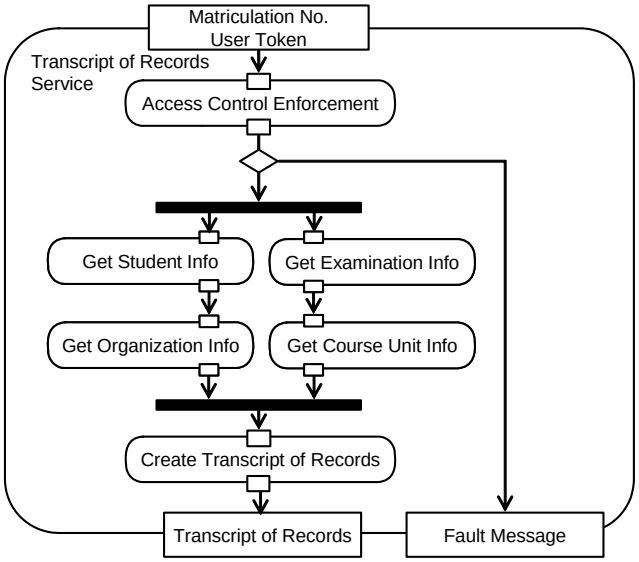
2.1.2 Erweiterung um Zugriffskontrolle

In der Analyse wurde gezeigt wie sich die Notwendigkeit einer Zugriffskontrolle von der Integrationsschicht, dem ToR-BPEL-Service auf die Basis-Webservices vererbt. Diese Vererbung ist erforderlich, um dem Leitprinzip der Webservice-Technologien, dem Prinzip der losen Kopplung, zu entsprechen und dabei zu verhindern, dass die Zugriffskontrolle des BPEL-Services umgangen werden kann. Der Kommunikationsschritt zwischen Wrapper-Komponente und Altsystem dagegen kann als „fest verdrahtet“ angesehen werden, so dass die Zugriffskontrollentscheidung nicht erneut gefällt werden muss.

Die Entscheidung, ob ein Benutzer, sei es ein menschlicher oder eine weitere Applikation, Zugriff erhält wird anhand eines identifizierenden Benutzertokens getroffen. Dieses könnte beispielsweise bei einer vorangegangenen Authentifizierung des Benutzers generiert worden sein. Genaue Vorbedingungen wie die erfolgreiche Authentifizierung, das Format des Benutzertokens oder weitere involvierte Komponenten wie beispielsweise ein *Decision Point* werden hier bewusst ausgeklammert, da sie sich außerhalb des Sichtbarkeitsbereiches dieser Architektur befinden. Die einzige zwar auch nur als Blackbox sichtbare Komponente der Zugriffskontrollinfrastruktur, sozusagen der Berührungspunkt zwischen funktionalen und nichtfunktionalen Komponenten, ist der *Access Control Enforcement Point*. Bei einer eingehenden Anfrage wird das mit der Anfrage mitgelieferte Benutzertoken zusammen mit einer Kennung für die betreffende Anforderung an den *Enforcement Point* weitergereicht. Die Antwort beinhaltet dann die jeweilige Entscheidung.

WS-Security bringt zwar einen Transportmechanismus für Benutzertoken mit, allerdings liegt es dann in der Verantwortung des Application Servers was damit geschieht. Viele WS-Security-Implementierungen unterstützen nur den Transport des *Username Tokens* zur Identifizierung bzw. Authentifizierung des Senders (oft: *wsse authentication*), dabei „verschwindet" dieses Token schnell in der jeweils serverspezifischen Umsetzung des Java Authentication and Authorization Services (JAAS). Eine auf diesem Mechanismus basierende Lösung wäre bei Verwendung der vorgefundenen WS-Security-Implementierungen zumindest serverabhängig und damit nicht ohne weiteres portabel. Zum anderen würde bei Verwendung dieses Transportmechanismus die Durchsetzung der Zugriffskontrollentscheidung von der betreffenden Komponente weg zum WS-Security-Handler gelegt werden, im Falle eines WS-Security-Gateways also sogar in einen anderen Serverprozess. Es ist aber in jedem Fall wünschenswert die Zugriffskontrollentscheidung transparent in der Nähe der betreffenden Ressource zu treffen, schon um eine gewisse Übersichtlichkeit und Nachvollziehbarkeit zu gewährleisten. Auf der anderen Seite ist es natürlich wünschenswert, durch Hinzunehmen eines Zugriffskontrollmechanismus die fachfunktionalen Komponenten nicht zu verändern, das Benutzertoken also außerhalb der eigentlichen Nutzlast einer Nachricht zu transportieren.

Daher wird der Entwurf aus [Sc06] für die Verzahnung der Zugriffskontrollmechanismen mit den fachfunktionalen Komponenten als Kompromiss zwischen dem Prinzip *separation of concerns* und dem Prinzip der Plattformunabhängigkeit vorgeschlagen. Ein analoges Vorgehen für den Aufruf des *Enforcement Points* wird auch für die Basis-Webservices empfohlen:



Information 19: Transcript of Records Service with Access Control Enforcement

(1) Als weiterer Eingabeparameter wird ein Benutzertoken, zum Beispiel eine einfache Identifikationsnummer, erwartet.

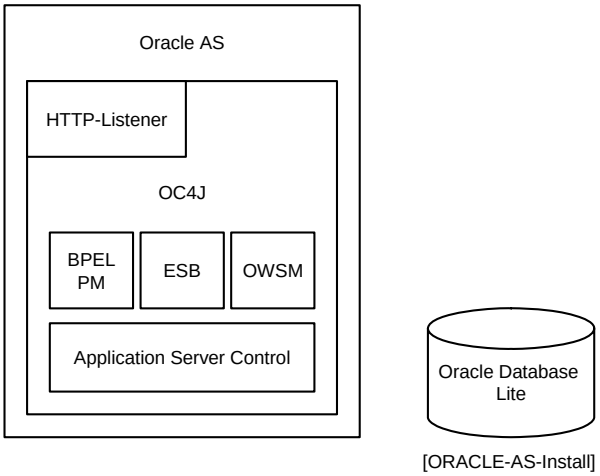
(2) Hinter der Aktivität „Access Control Enforcement" steckt ein Aufruf an den *Access Control Enforcement Point*. In diesem Aufruf wird das Benutzertoken zusammen mit einer Dienstkennung übergeben. Die Antwort enthält die Zugriffskontrollentscheidung. Nur bei positiver Entscheidung wird der Prozess ausgeführt, im anderen Fall wird ein Abbruch herbeigeführt.

2.2 Architektur auf Dienstkompositionsebene

Information 18 zeigt unter anderem bereits alle Komponenten die auf der Kompositionsschicht für die Absicherung mit WS-Security benötigt werden. Dieses Kapitel skizziert deren Verteilung auf verschiedene Systeme und beschreibt die Beziehungen dieser Komponenten und Systeme untereinander.

2.2.1 Oracle SOA Suite

Die Oracle SOA Suite bietet eine umfassende Zusammenstellung von Infrastrukturkomponenten zur Unterstützung bei Entwicklung, Betrieb und Management einer Webservice-basierten Service-orientierten Architektur.



Information 20: Oracle SOA Suite - Basic Installation Topology

Eine Basis-Installation der Oracle SOA Suite besteht aus folgenden Komponenten:

(1) Der Oracle Application Server stellt die Basisfunktionalität des Servers bereit. Je nach Bedarf können *Web Container* und *J2EE Container* integriert werden. Unterstützung bei Betrieb und Verwaltung stellt der Oracle Enterprise Manager zusammen mit einer Webbrowser-basierten Benutzerschnittstelle (Application Server Control) zur Verfügung. Zur Infrastruktur gehört außerdem der Oracle Process Manager and Notification Server (OPMN).

(2) Der eigentliche *J2EE Container* OC4J ist vollständig in Java implementiert und kann auf einer gewöhnlichen Java Virtual Machine (JVM) ausgeführt werden. Er besitzt neben einem eigenen HTTP-Listener einen *JSP Translator*, eine *Servlet Engine* und einen Container für Enterprise JavaBeans (EJB). Der OC4J wird hier lediglich zum Betrieb des BPEL Process Managers und des Web Service Managers benötigt.

(3) Der BPEL Process Manager stellt sämtliche Funktionalität zum Betrieb von BPEL-Prozessen zur Verfügung. Eine nähere Beschreibung des Aufbaus liefert Kapitel 2.2.2.

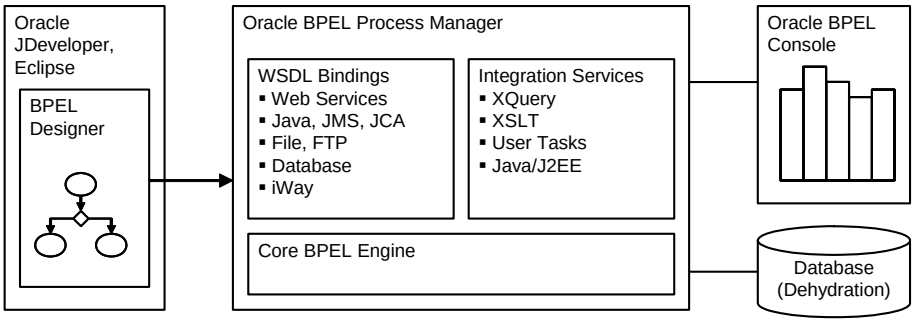
(4) Der Oracle Web Services Manager (OWSM) erweitert den Application Server bzw. den *J2EE Container* um Sicherheitsfunktionalität. Eine umfassendere Beschreibung liefert Kapitel 2.2.3.

(5) Über die Application Server Control Console sind Produktdokumentationen und weitere Managementkonsolen wie die Oracle BPEL Console oder die OWSM Control zugänglich.

(6) Die Olite Datenbank wird z.B. vom OWSM als *Policy Store* verwendet.

2.2.2 Oracle BPEL Process Manager

Für den Betrieb des BPEL-Prozesses soll der Oracle BPEL Process Manager genutzt werden.



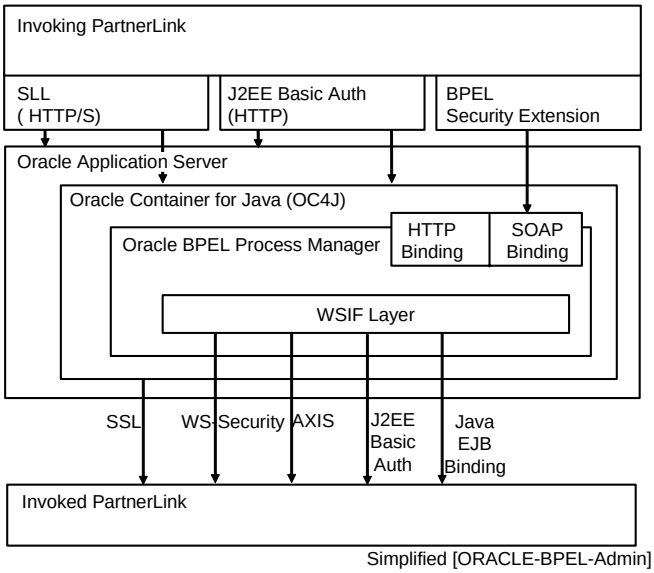
Information 21: Oracle BPEL Process Manager Architecture

(1) Zur Entwicklung von BPEL-Prozessen für den Oracle BPEL Process Manager stellt Oracle den BPEL Designer für die hauseigene Entwicklungsumgebung JDeveloper und die Open-Source IDE Eclipse zur Verfügung.

(2) Den Betrieb von BPEL-Prozessen erledigt der Oracle BPEL Process Manager. Dieser benötigt selbst einen J2EE Application Server, also beispielsweise den Oracle AS mit integriertem Oracle Container for Java (OC4J). Alternativ wären auch andere Application Server geeignet, darunter der JBoss AS oder IBM WebSphere. Der BPEL Process Manager selbst basiert auf einer BPEL-Engine und verwaltet daneben weitere Integrationsdienste wie eXtensible Stylesheet Languages Transformation (XSLT) oder zur einen Dienst zur Ausführung von Java-Programmcode innerhalb des Prozesses. Weiter unterstützt er verschiedene Schnittstellen zur Ansteuerung via WSDLs, dazu gehören unter anderen SOAP und der Java Messaging Service (JMS). Allerdings besitzt der Process Manager keine interne SOAP-Engine sondern bedient sich eines SOAP-Gateways. Die *Core BPEL Engine* besteht selbst aus weiteren internen Komponenten, die hier nicht dargestellt sind wie der *Lifecycle Manager*, der verschiedene Versionen eines Projektes verwaltet, ein Persistenzdienst, der den aktuellen Zustand eines Prozesses in eine externe Datenbank schreiben kann und einer WSIF-Komponente.

(3) Zur Verwaltung von langlebigen BPEL-Prozessen kann eine externe Datenbank genutzt werde, diese hält den zuletzt eingenommen Zustand des BPEL-Prozesses persistent vor. Oracle bezeichnet diesen Vorgang mit *Deydration*.

(4) Die Oracle BPEL Console dient zur Überwachung von BPEL-Prozessen während des Betriebs.



Simplified [ORACLE-BPEL-Admin]

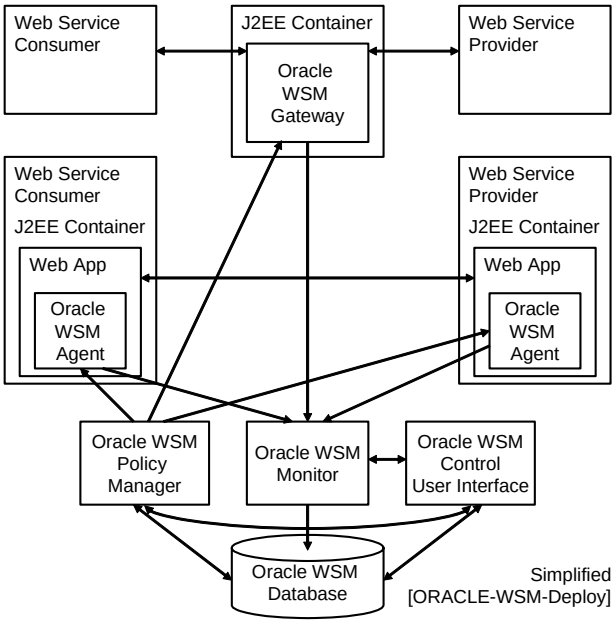
Information 22: Securing BPEL Processes with Oracle BPEL Process Manager

(1) Die Absicherung eines BPEL-Prozesses kann auf verschiedenen Wegen geschehen. Dabei stehen der Standard-Konfiguration, wie sie in Information 22 gezeigt wird (Oracle AS, OC4J, BPEL PM), vor allem Transportlevel-Absicherungsmechanismen wie HTTP/S und HTTP Basic Authentication zur Verfügung. Je nach Installationsart des BPEL PMs übernimmt der Application Server oder der Container die Behandlung von Authentifizierung und Verschlüsselung.

(2) Ausgehende Aufrufe werden generell vom Container mittels SSL abgesichert oder was die Authentisierung betrifft von der BPEL PM-internen Web Service Invocation Layer (WSIF)-Komponente. Hier stehen verschiedene Spielarten zur Verfügung, neben anderen WS-Security, eingeschränkt auf das *Username Token* und HTTP Basic Authentication. Unterstützung wird weder für XML Encryption noch für XML-Signature geboten. Zur Verschlüsselung stehen damit ausschließlich Transportlevel-Mechanismen zur Verfügung.

2.2.3 Oracle Web Services Manager

Zur Absicherung der Webservice-Interaktionen wird daher eine zusätzliche Komponente benötigt. Der Oracle Web Services Manager (OWSM) bringt diese zusätzliche Funktionalität ein.



Information 23: Securing Web Services with Oracle Web Services Manager

Dabei handelt es sich um eine Policy-basierte Infrastruktur bestehend aus:

(1) *Enforcement Points*, welche die Nachrichten zwischen Webservice-Verbraucher und Webservice-Anbieter abfangen und für die Durchsetzung der entsprechenden Policies verantwortlich sind. Zwei Ausprägungen dieser *Enforcement Points* werden zur Verfügung gestellt, zum einen als *Agent* und zum anderen als *Gateway*. Ein *Gateway* agiert als eigenständiger Serverprozess, ein *Agent* wird innerhalb des Containers ausgeführt, in welchem sich auch die Webservice-Anwendung befindet. Die Kommunikation zwischen *Enforcement Points* und andern OWSM-Komponenten geschieht via RMI.

(2) Eine weitere Komponente ist der OWSM Policy Manager, der die *Agents* und *Gateways* verwaltet und steuert.

(3) Der OWSM Monitor dient der Überwachung von *Agents* und *Gateways*.

(4) Die Komponente OWSM Control ist die Schnittstelle zwischen OWSM Policy Manager, OWSM Monitor und Administrator.

(5) Die OWSM-Datenbank dient unter anderem als *Policy Store*. Die Kommunikation zwischen Datenbank und den anderen OWSM Komponenten geschieht via JDBC.

Neben weiteren Funktionen können mit dem Oracle WSM vor allem *Security Policies* zur Absicherung von Webservices mit WS-Security erstellt und durchgesetzt werden. Zur Absicherung eines BPEL-Prozesses kommt nur das OWSM Gateway als *Enforcement Point* in Frage.

(1) Keystore types
  (1) Java Keystore (jks)
  (2) PKCS12

(2) XML-Signature
  (1) RSA-SHA 1
  (2) DSA-SHA 1

(3) XML Encryption
  (1) 3DES
  (2) AES-128
  (3) AES-256

Information 24: Signature and Encryption with OWSM

Folgende Konfigurationsmöglichkeiten für das Signieren und Verschlüsseln werden vom OSWM bereitgestellt:

(1) Es werden Keystores zur Verschlüsselung wie zur Signierung vom Typ „jks“ und vom Typ „PKCS12“ unterstützt.

(2) Zur Berechnung von Signaturen stehen zwei Verfahren zur Auswahl RSA mit SHA 1 und DSA mit SHA 1.

(3) Zur Verschlüsselung können der TripleDES-Algorithmus oder der AES-Algorithmus mit 128 bzw. 256 Bit Schlüssellänge verwendet werden.

Information 25 zeigt einen Ausschnitt aus einer beispielhaften (auf das Wesentliche reduzierten) *Security Policy*, wie sie mit Hilfe der graphischen Benutzeroberfläche des OWSM erzeugt werden kann.

```
<wsp:Policy name="StudentDBService" >
   <csw:Pipeline type="Request">
      <csw:Step name="Sign Message and Encrypt">
         <csw:Properties>
            <csw:PropertySet name="Signing Properties">
               <csw:Property name="SignerKeyStoreFile">c:\client.keystore</csw:Property>
               <csw:Property name="SignerKeyStoreType">jks</csw:Property>
               <csw:Property name="SignerKeyStorePassword">????BeBnZDOK...</csw:Property>
               <csw:Property name="SignerKeyAlias">cmtm</csw:Property>
               <csw:Property name="SignatureAlgorithm">RSA-SHA1</csw:Property>
               <csw:Property name="SignedContent">BODY</csw:Property>
            </csw:PropertySet>
            <csw:PropertySet name="Encryption Properties">
               <csw:Property name="EncryptKeyStoreFile">c:\client.keystore</csw:Property>
               <csw:Property name="EncryptKeyStoreType">jks</csw:Property>
               <csw:Property name="EncryptKeyStorePassword">????BS1e2knKH...</csw:Property>
               <csw:Property name="EncryptKeyAlias">cmtm</csw:Property>
               <csw:Property name="EncryptionAlgorithm">AES-128</csw:Property>
               <csw:Property name="KeyTransportAlgorithm">RSA-1_5</csw:Property>
               <csw:Property name="EncryptedContent">BODY</csw:Property>
            </csw:PropertySet>
         </csw:Properties>
      </csw:Step>
   </csw:Pipeline>
</wsp:Policy>
```

Information 25: Exemplary OWSM Security Policy

Beim Aufruf des StudentDBServices soll die zu sendende Nachricht signiert und anschließend verschlüsselt werden.

(1) Zur Signierung ist die Angabe eines Keystores zusammen mit Typ, Passwort und Alias des zu verwendenden Schlüssels notwendig. Hier wurde der Signierungsalgorithmus RSA mit Digest SHA 1 gewählt. Die Signatur wird für den gesamten Body der SOAP-Nachricht erstellt.

(2) Auch zur Verschlüsselung werden Angaben bezüglich eines Keystores benötigt. Verschlüsselt wird dann mit dem AES-128-Verfahren. Da dies ein symmetrisches Verfahren ist, wird der entsprechende geheime Schlüssel zunächst asymmetrisch verschlüsselt, hier mit RSA 1.5, und dann in der Nachricht mit gesendet. Die Verschlüsselung umfasst hier den gesamten Body der SOAP-Nachricht. Zur Verschlüsselung ausgewählter Nachrichtenteile währen auch XPath-Angaben möglich.

2.3 Architektur auf Ebene der Basis-Webservices und der angekoppelten Altsysteme

JBoss AS (JBoss Application Server) ist ein populärer Open-Source-Java-Applikation-Server im Markt. JBoss stellt einen vollständig J2EE-basierten Applikationsserver für Unternehmen zur Verfügung. Beim Einsatz in Unternehmen spielen verteilte Anwendungen eine besonders große Rolle. Um einen Überblick über JBoss AS zu erhalten, wird dieses Kapitel wie folgt untergliedert.

(1) J2EE

(2) JBoss AS Architecture

(3) JBossWS

(4) JBossWS and WS-Security

(5) JBoss AS and SAP CM

Information 26: JBoss Application Server and Legacy Systems - Content Overview

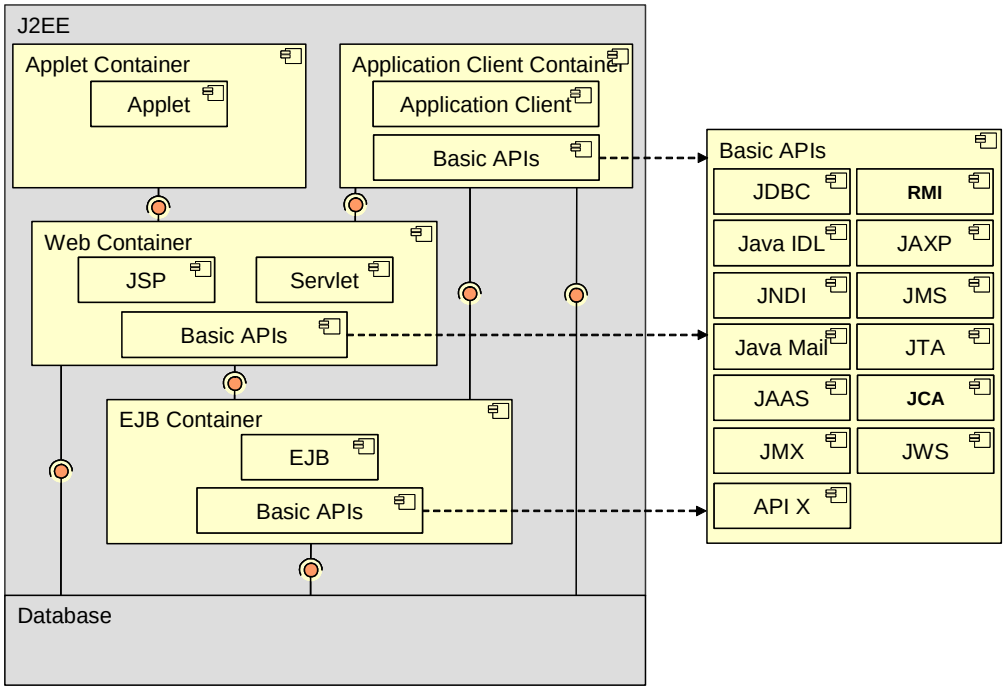
Zunächst werden im folgenden Abschnitt die Funktionalitäten von J2EE vorgestellt. Danach steht die JBoss AS-Architektur im Mittelpunkt. Es handelt sich dabei um die Microkernstruktur und die Hauptmodule der internen JBoss AS-Architektur. Der nächste Abschnitt beschreibt die Webservice-Infrastruktur des JBoss AS. Da die Sicherheitsbehandlung von SOAP-Nachrichten bei Verwendung durch Webservice-Anwendungen auf J2EE-Applikation-Servern immer noch nicht einheitlich integriert ist, wird der OASIS-Standard Web Services Security (WSS) auf dem JBoss AS verwendet, um diese Sicherheitslücke zu schließen. Zum Schluss wird die Kommunikation zwischen JBoss AS und SAP CM zur Bereitstellung von Webservice-Schnittstellen skizziert.

2.3.1 J2EE - Überblick

Java 2 Platform Enterprise Edition (J2EE) definiert eine Java-Plattform für transaktionsbasierte Unternehmensanwendungen. Das J2EE-Rahmenwerk arbeitet mit Basiskomponenten, mit denen Anwendungen verfügbar gemacht werden können. Wenn eine Anwendung J2EE-konform ist, dann kann sie auf einen anderen J2EE-Applikation-Server portiert werden. Die J2EE-Plattform ist eine Kollektion der Enterprise-APIs und ein Framework für auf Komponenten basierte Unternehmensrechner.

J2EE-Architektur

Die J2EE-API besteht aus folgenden Funktionsbausteinen (siehe Information 27):



Information 27: J2EE Architecture Overview

(JDBC) *Java Database Connectivity* dient als eine einheitliche Schnittstelle zur Erstellung einer Verbindung mit den relationalen Datenbanken verschiedener Hersteller. Für jede spezifische Datenbank werden eigene Treiber nach der JDBC-Spezifikation implementiert. JDBC ermöglicht, Datenbankverbindungen aufzubauen und zu verwalten, SQL-Anfragen an die Datenbank weiterzuleiten und die Ergebnisse in eine für Java nutzbare Form umzuwandeln und die für die Anwendungen wieder zu verwenden.

(RMI) *Remote Method Invocation* erlaubt es einem Klient, sich mit einem Server zu verbinden und die Methoden der entfernten Java-Objekte auf dem Server aufzurufen. Wenn sich ein Objekt in einer anderen Java Virtuellen Maschine befinden kann, kann dieses Objekt prinzipiell auf einem entfernten Rechner oder auf dem lokalen Rechner laufen. Der Aufruf einer Methode eines entfernten Objektes funktioniert genauso wie ein lokaler Aufruf ohne merkbaren Unterschied. Auf der Client-Seite wird der Netzwerktransport vom Stub betreut. Der Stub muss entweder lokal oder über das Netz für den Client bereitgestellt werden.

(Java IDL) *Interface Definition Language* stellt eine Schnittstelle zu den entfernten Objekten zur Verfügung. Mithilfe dieser Schnittstellenbeschreibungssprache werden Objekte und Methoden mit möglichen Parametern und Datentypen beschrieben, ohne dabei die Eigenschaften einer bestimmten Programmiersprache zu verwenden. IDL befindet sich meist in verteilten Systemen, wobei ein Klient die Methoden der entfernen Objekte auf einem anderen Rechner ausführen kann, z.B. wie COM, CORBA oder SOAP.

(JAXP) *Java API for XML Parsing* ist eine Java-XML-API, die zur grammatischen Analyse und Benachrichtigung der XML dient. JAXP enthält zwei unterschiedliche Schnittstellen, nämlich *Document Object Model* (DOM) und *Simple API for XML* (SAX), um ein XML-Dokument oder einen XML-Abschnitt zu analysieren. Außerdem wird eine XSLT-Schnittstelle für Transformationen an Daten und Strukturen eines XML-Dokuments realisiert.

(JNDI) *Java Naming and Directory Interface* definiert eine Programmierschnittstelle für Namen- und Verzeichnisdienste, wie beispielsweise LDAP, DNS, NIS oder Novells NDS. Diese Schnittstelle hängt nicht von der tatsächlichen Implementierung ab. JNDI dient vor allem zur Registrierung der verteilten Objekte über Netzwerk und Aufrufe der entfernten Methoden in J2EE-Anwendungen.

(JMS) *Java Messaging Service* stellt eine Programmierschnittstelle für vernetzte Nachrichtenübermittelungsdienstleistungen und für das Schreiben der Nachricht-orientierten Middleware zur Verfügung. In J2EE-Anwendungen ist JMS verantwortlich für den Nachrichtenaustausch zwischen Java-Klienten.

(JavaMail) *JavaMail* dient zur Benachrichtigung im E-Mail-Format.

(JTA) *JavaTransaction API*: stellt eine Programmierschnittstelle für die Verwaltung der verteilten Transaktionen von mehreren Komponenten einer oder mehrerer Anwendungen zur Verfügung. JTA beschreibt einen Dienst, der alle Transaktionen zentral kontrollieren kann.

(JAAS) *Java Authentication and Authorization Service* kümmert sich um Authentifizierung und Zugriffsrechte in J2EE-Anwendungen. Die Architektur von JAAS wurde von den *Plugable Authentication Modules* (PAM) vom Unix-System beeinflusst. Dadurch unterstützt JAAS eine benutzerbasierte Autorisierung. JAAS bietet eine flexibele Authentifizierung über wesentliche einfache auswechselbare und kaskadierbare Anthentifizierungsmodule an. Single Sign-On bei den Anwendungen in einer J2EE-Plattform ist möglich durch JAAS. Schließlich können die Regeln in den zuvor beschriebenen Sicherheits-Policies an eine Benutzeridentität durch auf JAAS-basierte Autorisierungskomponenten verknüpft werden.

(JCA) *J2EE Connector Architecture* definiert eine Software-Architektur und Programmierschnittstelle für die Kommunikation zwischen Anwendungen auf der J2EE-Plattform. Die *Enterprise Application Integration* (EAI) beschreibt, wie bestehende Applikationen und Datenquellen integriert werden. In einer J2EE-Plattform lässt sich ein EAI aus einer oder mehreren Applikationskomponenten in einem entsprechenden Container ablaufen.

(JMX) *Java Management Extension* ist eine Spezifikation, die die Überwachung und Verwaltung der J2EE-Anwendungen beschreibt. JMX sind Kern-Bauelemente von JBoss AS. Diese werden im nächsten Abschnitt detailliert erläutert.

(JWS) *Java Web Services* ist eine Software-Anwendung, die durch einen *Uniform Resource Identifier* (URI) eindeutig identifiziert werden kann und Nachrichten im XML-Format mit anderen Software-Agenten über ein existierendes Internet-Protokoll wie beispielsweise HTTP oder SMTP austauschen kann. Dieser Teil wird ebenfalls im nächsten JBoss AS-Abschnitt beschrieben. Dabei dienen die (JAX-WS) *Java API for XML – Web Services* als Java API zum Erstellen von Webservice-Endpunkten und Klienten. JAX-WS Annotationen können von anderen J2EE APIs benötigt werden, um die Entwicklung und das Deployment von Webservice-Endpunkten und Klienten zu vereinfachen. (JAX-PRC) *Java API for XML-Based Remote Procedure Calls* bietet eine API-Schnittstelle zur XML-Kommunikation über SOAP-Protokoll an. Jede Methode hat ihr eigenes Kommunikationsprotokoll und ihr eigenes Nachrichtenformat. JAX-WS basiert auf JAX-RPC APIs. JSR-109 spezifiziert ein Programmiermodell für Enterprise Webservices mit Hilfe JAX-WS. Und JSR-181 definiert eine annotierte Java-Syntax zum Programmieren von Webservices. JSR-181 und JSR-109 dienen zur Implementierung von Enterprise Webservices zusammen.

J2EE-Komponentenmodelle

Außerdem gehören zum J2EE als ein Komponenten-Framework noch folgende APIs:

1. Enterprise JavaBean 2.1
   1. J2EE 1.4 application component
   2. Supports JSR-109 web service endpoints
2. Enterprise JavaBean 3
   1. Java EE 5 application component
   2. Supports JSR-109 and JSR-181 web service endpoints
3. Servlet e.g. Plain Old Java Object
   1. Java web (application) component
   2. Supports JSR-109 and JSR-181 web service endpoints
4. Java Server Pages
   1. Java web component
   2. No support for web service endpoints

Information 28: J2EE Component Models Supported by JBoss AS

(EJB) *Enterprise Java Beans*: definiert ein Komponenten-Modell für das Trennen der Einheit von Geschäftslogik und Datenlogik. Auf der J2EE-Plattform können EJBs wichtige von Geschäftslogik oder Datenlogik abhängige Konzepte für Unternehmensanwendungen wie beispielsweise Transaktions- Namens- oder Sicherheitsdienste standardisieren.

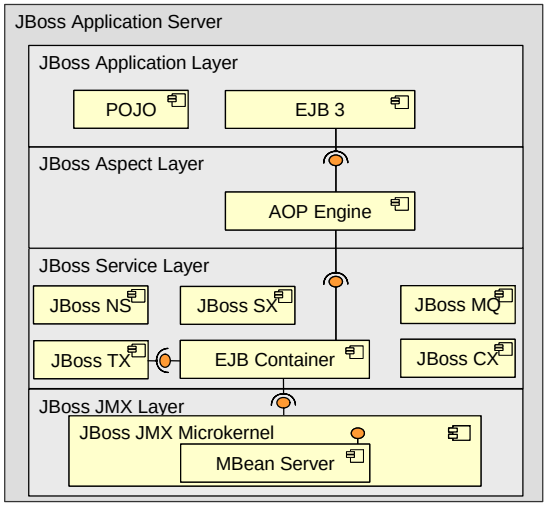
(Servlet) *Servlet*: ist eine Java-Klasse, deren Instanzen innerhalb eines J2EE-Applikationservers Anfragen von Klienten entgegen nehmen und beantworten kann. Der Inhalt der Antworten kann statisch oder dynamisch für den Webserver erstellt werden. Eine besondere Form des Servlets stellt die sogenannte Plain Old Java Object (POJO)-Komponente dar, mit einfachen Mitteln kann eine alte Java-Klasse als Webservice veröffentlicht werden.

(JSP) *Java Server Page*: dient vereinfacht zur dynamischen Erzeugung von HTML- und XML-Dokumenten oder –Abschnitten eines Webservers in der J2EE-Plattform. Die Verwendung der JSP-Technologie trennt Webseitendesign von Logik.

2.3.2 JBoss Appilcation Server

Der JBoss Application Server (JBoss AS) ist ein Open-Source-Application Server, der sämtliche J2EE-APIs standard-konform implementiert. JBoss AS wurde von *Marc Fleury* im Jahre 1999 entwickelt. Nach zwei Jahren wurde JBoss AS als Open-Source-AS veröffentlicht. Im Jahre 2004 wurde der JBoss AS J2EE 1.4-zertifiziert. Unter der LGPL-Lizenz kann JBoss AS inklusive Quelltext kostenlos bezogen und frei verwendet werden.

Zu Beginn dieses Abschnitts wird die JMX-Infrastruktur vorgestellt. Die JBoss AS-Architektur fußt hauptsächlich auf dieser JMX-Infrastruktur, im Vergleich mit anderen J2EE-AS wurde der JMX-Standard hier wesentlich umfassender umgesetzt. Anschließend werden die Hauptmodule der JBoss AS-Architektur behandelt.



Information 29: JBoss AS Architecture

Wie in Information 29 erkennbar, besteht der JBoss AS im Allgemeinen aus vier Layern:

(*JBoss JMX Layer*): JMX Microkernel stellt Funktionalitäten eines von Hot-Deployment gestützten Komponentenmodells mit dynamischen class-loading-Eigenschaften und Lebenszyklusmanagement zur Verfügung.

(*JBoss Service Layer*): dieser Layer ermöglicht es alle zusätzlichen Dienste wie Transaktionsservices, Messaging-Services oder Sicherheitsservices als Plug-In-Service in JBoss AS zu nutzen. Jeder Service wird als eine von Hot-Deployment gestützten Komponenten mit dem Name „*Service Archive*“ (SAR) integriert.

(*JBoss Aspect Layer*): Eine so genannte *Aspect Oriented Programming* (AOP) Engine verknüpft Servicekomponenten mit Interceptors, die für Ermittlung der Service-Behavior in Objekten zuständig sind.

(*JBoss Application Layer*): Applikationen für Endbenutzer werden auf diesem Layer erstellt. Siehe hierzu Abbildung 26.

JBoss AS hat folgende Eigenschaften:

(1) Skalabilität: JBoss AS unterstützt alle auf Java-Technologie basierten Objekte wie EJB, POJO.

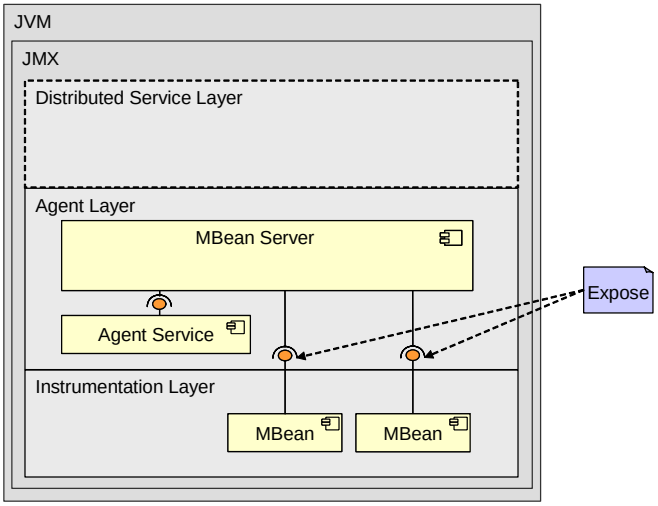
(2) Modulare Architektur: JBoss AS bietet eine einzigartige Architektur an, die auf einem Mikrokernel-basierten Design errichtet wurde, welches die Java Management-Erweiterungen (Java Management Extensions - JMX Technologie) wirksam einsetzt. Das Ergebnis ist ein schlankes Komponenten-Modell, das class-loading Eigenschaften und Management liefert.

(3) SOA: Services sind leicht einzufügen oder zu entfernen nach eigenen spezifischen Benutzeranforderungen. Alle Services sind sorgfältig implementiert und unterstützen vollständig hot-deployable.

JBoss JMX Layer

Wie im letzten J2EE-Abschnitt erläutert, standardisiert JMX Java Anwendungen und Dienste. JMX stellt eine durch den Java Community Process entwickelte Spezifikation zur Verfügung, die im Vergleich zu vielen anderen Java-Technologie-Ansätzen Architektur, Design Patterns,

eine API, Verwaltungsdienste für Anwendungen und Netzwerke sowie Monitoring Dienste für Java beschreibt. JMX erleichtert die Kommunikation zwischen unterschiedlichen Java-Anwendungen innerhalb einer oder mehrerer JVM (ab Java Version 6). Diese Erleichterung wird durch die Unterstützung von Adapter und Connectoren ermöglicht. Ein HTTP-Adapter kann implementiert werden, um alle Anwendungen über einen Web-Browser leicht steuerbar zu machen (Information 30).



Information 30: JMX - Layered Model

MBeanServer in der JBoss-Architektur spielt eine Rolle als Microkernel-Aggregatorkomponente. Alle anderen MBeans-Komponenten sind dynamisch auf MBeanServer eingefügt. Der Kern ist keine echte Funktionalität, sondern ein Aggregator.

Jede konforme Anwendung kann im Prinzip über die einheitliche JMX-Schnittstelle von jeder JMX-fähigen Management Konsole überwacht, entdeckt und verwaltet werden. JMX ist ein wieder verwendbares Framework, das die Applikationen für die lokalen oder entfernten Managementwerkzeuge aufgeben kann. JMX besteht aus Instrumentations-, Agent- und Verteilungsdienstebenen.

(1) Die Instrumentierungsebene stattet die Ressourcen mit den Eigenschaften aus und fördert komfortable JMX-Applikationen. Auf dieser Ebene befinden sich alle zu verwaltenden Ressourcen. Eine JMX-durchführbare Ressource kann alles sein wie beispielsweise Applikationen, Dienstkomponenten oder Geräte usw. Diese Ressource kann durch ein Java-Objekt oder einen Wrapper, der ihre Eigenschaft beschreibt, repräsentiert werden. Benutzer können die Eigenschaften einer gegebenen Ressource durch die Anwendung eines MBeans (sogenannter *Managed Bean*) realisieren. Die Intrumentationsebene ist nach einem Notifikationsmechanismus spezifiziert. Dieser Notifikationsmechanismus erlaubt, die Verbindung von MBeans innerhalb ihrer Laufzeitumgebung zu erstellen.

(2) Die Agentenebene ist verantwortlich für die Kontrolle und Ermittlung der zu verwaltenden Ressourcen, die mit dem Agent auf dem MBeanServer registriert werden. Auf dieser Ebene kann ein standardisierter MBeanServer-Management-Agent alle Objekte auf der Instrumentationsebene kontrollieren. Dabei werden Unterstützungsdienste und ein Kommunikationsvermittler auf dieser Ebene definiert. Der JMX-Agent kann in der Hardware, wo die Ressourcen stehen und wo eine *Java Virtual Machine* (JVM) läuft, lokalisiert werden. Es ist nicht notwendig für einen JMX-Agent zu wissen, für welche Ressourcen er benötigt wird. Im Gegensatz dazu können Ressourcen die benötigten Agenten in der JMX-Infrastuktur verwenden.

(3) Die Dokumentation für die Verteilungsdienstebene wird zurzeit für die kommende Spezifikation erwartet. Diese Ebene soll Schnittstellen definieren, die zur Implementierung der JMX-Management-Applikationen oder Verwalter notwendig sind.

JMX Bestandteile auf Instrumentierungsebene

Die Instrumentierungsebene besteht aus folgenden Bestandteilen:

(1) MBeans
  (1) Standard MBeans
  (2) Dynamic MBeans
  (3) Open MBeans
  (4) Model MBeans

(2) Notificationsmodel

(3) MBean Metadata

Information 31: Components Overview of JMX Instrumentation Layer

(1) *MBeans* ist ein reines Java-Objekt, das eine verwaltbare Ressource darstellt. MBeans beschreibt alle wichtigen Informationen über eine Ressource und Operationen, die zur Verwaltung einer Management-Applikation benötigt werden. Entsprechend ihrer Instrumentationsanforderungen können MBeans in vier Typen unterschieden werden:

(1.1) *Standard MBeans*: definieren eine einfache JavaBean und eine statische Managementschnittstelle mit der gleichen Bezeichnung.

(1.2) *Dynamic MBeans*: werden während der Laufzeit instanziiert oder verändert. Standard MBeans können mit Hilfe von Annotation zu Dynamic MBeans umgestellt werden.

(1.3) *Open MBeans*: sind Erweiterungen von Dynamic MBeans. Die stellen nur die Standard-Java-Typen zur Verfügung. Durch Open MBeans kann JMX mit anderen Managementprogrammen besser kommunizieren.

(1.4) *Model MBeans*: sind auch Erweiterungen von Dynamic MBeans mit zusätzlichen „Read“ und „Write“-Methoden. Mit den Methoden können Werte in einer Datei oder in einer Datenbank gespeichert oder ausgelesen werden. Model MBeans können direkt im Agent erzeugt werden.

(2) Notifikationsmodell: der JMX-Benachrichtigungsmechanismus basiert auf dem Java-Event-Mechanismus. Notifikationen sind ein Teil der Managementschnittstellen von MBeans. Ein so genannter Broadcaster MBeans ist zuständig für Ermittlung der Notifikationen. Der Notifikation-Listener bietet eine Empfang-Event-Schnittstelle an. Außerdem werden Operationen zur Registrierung der Notifikation-Listener auf dem MBean-Server definiert.

(3) *MBean Metadata*: MBean Metadata Klassen enthalten detaillierte Metadata-Informationen des selektierten MBean-Objektes.

JMX Bestandteile auf Agentenebene

Die Komponenten auf der Agentenebene beinhalten:

(1) MBean Server

(2) Agent Service
  (1) Dynamic Class Loading
  (2) Monitoring
  (3) Timer
  (4) Relations

Information 32: Components Overview of JMX Agent Layer

(1) MBean Server ist die Kernkomponente der gesamten JMX-Infrastruktur. MBeanServer beobachtet MBeans, konfiguriert die Arbeitswarteschlange, lädt MBeans von entfernten URLs dynamisch und definiert Verknüpfungen und Rollen zwischen MBeans für Abfragen und die Event-Verteilung. Auf dem MBeanServer kann eine MBean durch eine andere MBean, den Agenten oder eine entfernte Managementanwendung über verteilte Dienste mit einem eindeutigen Objektname instanziiert und angemeldet werden. Dabei dient der MBean-Server für MBean für die Ermittlung der entsprechenden Managementschnittstellen, zum Einlesen oder Schreiben der Attribute, zur Ausführung der von MBeans definierten Methoden, zur Registrierung des Notifikationsevents oder zur Anfrage von MBeans nach ihrem Objektnamen oder Attributen. Außer JVM werden vom Agenten Protokolladapter und Kommunikations-vermittler zur Verbindung mit MBean-Server eingefordert. Jeder Adapter kann alle angemeldeten MBeans vom MBean-Server auflisten, ein HTML-Adapter kann z.B. MBeans für die Modifizierung über Webbrowser verwenden. Der Kommunikationsvermittler (*Connector*) gilt als Programmierschnittstelle zur Erstellung einer vom Kommunikationsprotokoll unabhängigen Verbindung mit dem MBean-Server. Über unterschiedliche Protokolle kann jeder Vermittler identische entfernte Schnittstellen produzieren. Diese Eigenschaft ermöglicht es, eine fremde Managementanwendung mit einem bekanntgegebenen Agent über Netzwerk transparent zu verbinden ohne Berücksichtigung der gewendeten Protokolle. Adapter und Vermittler stellen alle Funktionalitäten vom lokalen MBean-Server für eine Managementanwendung bereit.

(2) Agentdienste sind zuständig für den externen Zugriff auf den Server über einen Kommunikationsadapter oder einen Connector. Agentdienste können angemeldete MBeans auf dem MBean Server verwalten. Normalerweise können Agentdienste genau wie MBeans ihre eigenen Funktionalitäten durch MBean Server kontrollieren. Die JMX Spezifikation definiert die folgenden Agentdienste:

(2.1) *Dynamic Class Loading*: durch ein (m-let) management applet service können die neuen Klassen und Bibliotheken vom Netzwerk dynamisch wiedergeholt oder instanziiert werden.

(2.2) *Monitoring*: die numerischen oder String-Werte von allen Attributen der MBeans können von Monitoren bei der Wertänderung beobachtet werden.

(2.3) *Timer*: ein Planerstellungsmechanismus wird verwendet und kann Notifikationen in einem vordefinierten Intervall automatisch verschicken.

(2.4) *Relations*: der Beziehungsdienst definiert Assoziationen zwischen MBeans und gewährleistet die Konsistenz dieser Beziehung.

JBoss Service Layer

Der JBoss Service Layer ist ein vom MBean-Server verbundenes MBeans (Management-Beans). Jedes Modul kann auf JMX-Microkernel durch Modifizierung der JBoss AS-Konfigurationen adaptiert oder angeglichen werden.

(1) JBoss EJB-Container

(2) JBoss Naming Service

(3) JBoss Transaction Service

(4) Deployment Service

(5) JBoss Messaging

(6) JBoss Security Service

(7) JBoss Connector Service

(8) JBoss Web

Information 33: JBoss Services Overview

Die JBoss-Hauptmodule auf dem JBoss Service Layer können wie folgt aufgeteilt werden:

(*JBoss EJB-Container*) ist die Kernimplementierung eines JBoss AS. Der EJB-Container generiert stub- und skeleton-Klassen vom EJB-Objekt in der Laufzeitumgebung und unterstützt hot-deployment oder hot-redeployment. In JBoss wird ein EJB-Container-Objekt als eine Unterklasse vom Container instanziiert. EJBDeployer-MBean verwaltet die Erstellung dieses Objektes.

(*JBossNS*) *JBoss Naming Service*: wird nach JNDI-J2EE-Spezifikation für die Lokalisierung der Objekte und Ressourcen implementiert. Die JBossNS-Architektur basiert auf der Java-Socket/RMI javax.naming.Context Schnittstelle. Die Implementierung ist optimiert, so dass die Ausführung einer fremden Ressource keine Sockets innerhalb der gleichen JVM, wo der JBossNS-Server läuft, verwenden kann.

(*JBossTX*) *JBoss Transaction Service* ist ein von Java Transaction API (JTA)/JTS gestützter Transaktionsmonitor. JBossTX-Architektur erlaubt, alle JTA-fähigen Transaktionsmanager zu implementieren. JBossTX enthält eine fast-in-VM Implementierung eines JTA-Transaktionsmanager, der als Default-Transaktionsmanager benutzt wird.

(*Deployment Service*) unterstützt die Inbetriebnahme der verschiedenen Komponenten, von EJB-Pakete (*java archive* JAR), Webapplikationen (*web application archive* WAR) und Enterprise-Applikationen (*enterprise application archive* EAR). Der Deployment-Service überwacht URLs für das J2EE-Archiv und nimmt die Archive bei Verwendung oder Veränderungen ab sofort in Betrieb.

(*JBossMQ*) *JBoss Messaging* ist eine Implementierung der *Java Messaging Specification* (JMS). JBossMQ besteht aus einigen Diensten, die JMS-API-Level-Dienste für Klientanwendungen zur Verfügung zusammenstellen können.

(*JBossSX*) *JBoss Security Service* unterstützt nicht-JAAS oder JAAS Sicherheits-implementierungen. Über einen Security-Proxy-Layer können Kundensicherheitsanforderungen integriert werden, ohne das Geschäftsobjekt von EJB zu verändern.

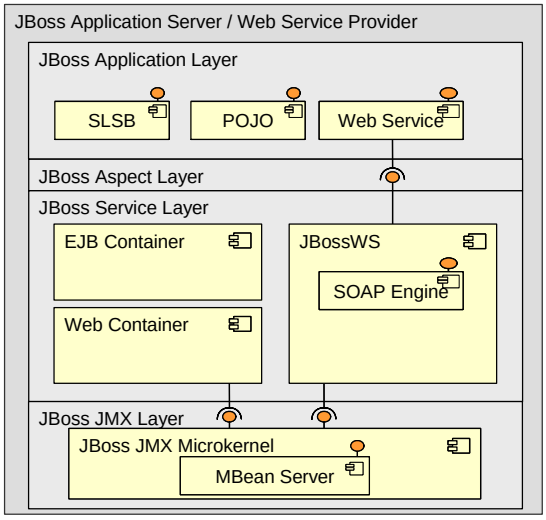
(*JBossCX*) *JBoss Connector Service* wird nach JCA-Spezifiaktion implementiert. JBossCX bietet einen Connection-Pool für JBoss AS zu Verbindungen mit JCA-Ressourcenadaptern an. Dabei ist ein Connection-Manager zuständig für die Verwaltung dieser Verbindungen.

(*JBoss Web*) Web Servers sind verantwortlich für Web-Container und Servlet-Engine.

Wenn ein JBoss AS gestartet wird, wird eine MBean-Instanz auf dem MBean-Server erstellt. Die MBean-Komponenten werden im JBoss AS durch Registrierung auf dem MBean-Server eingesetzt. JBoss AS implementiert die dynamische Klassenladen-M-Let-Services, die grundsätzlich ein Agent-Dienst ist. Dieser Dienst erlaubt MBeans sich auf dem MBeans-Server zu registrieren. Die zu ladenden MBeans sind durch ein Konfigurationsdokument im Text-Format spezifiziert.

Auf dem JMX-MBean-Server werden tatsächlich keine weiteren Funktionalitäten ausgeführt. Der JMX-MBean-Server spielt die Rolle als Dienstleister (Aggregator)-Komponente in der micro-Kern-Architektur, der zuständig für die Interkommunikation zwischen MBeans-Komponenten ist. Die entsprechenden Funktionalitäten werden von MBeans statt JMX-MBeans erbracht. Die allgemeine Architektur vom JBoss AS kann nicht so streng kategorisiert werden. Wie bereits erwähnt, ist JBoss AS ein Framework, auf dem Komponenten durch plug-in ergänzt oder entfernt werden können. Die Schnittstelle zwischen MBeans ist der Vermittler.

2.3.3 JBossWS - JBoss Web Services

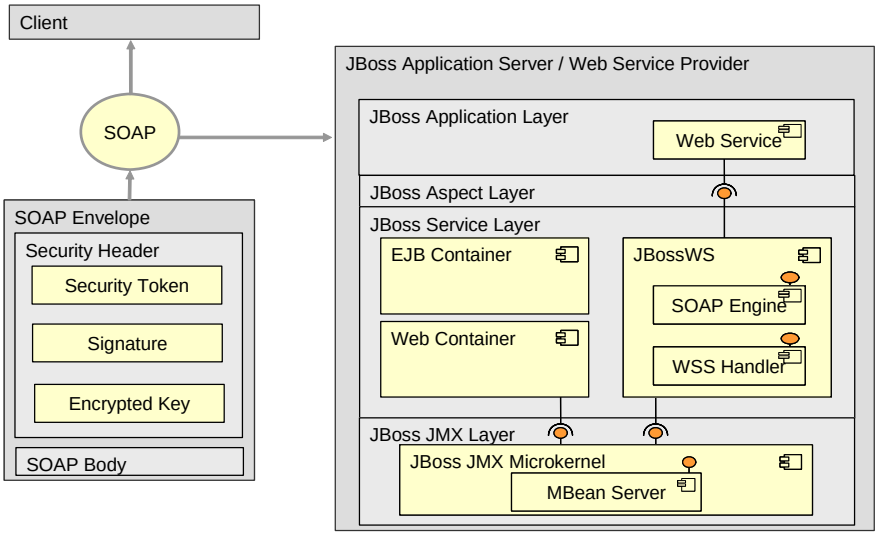


Information 34: JBossWS and JBoss AS

Wie Information 34 zeigt, wird JBoss Web Services (JBossWS) ebenfalls als ein Plug-In-Service im JBoss AS in Betrieb genommen. Zuerst wird JBossWS beim JMX MBean Server angemeldet und von ihm während der Laufzeit verwaltet. JBowss-WS unterstützt zwei Komponententypen, nämlich Stateless Session Beans (SLSB) und *Plain Old Java Objects* (POJOs). Session-Beans werden im EJB-Container, POJOs im Web-Container betrieben. JBossWS präsentiert diese als Webservice, die Konfiguration gelingt entweder anhand von Deployment-Deskriptoren im jeweiligen Container oder über JSR-181-Annotationen, die im Aspekt-Layer ausgewertet werden. JBossWS enthält unter anderem eine SOAP-Engine, die verantwortlich für das Senden und Empfangen von SOAP-Nachrichten ist. Zur Übersetzung von Java-Objekten in XML-Dokumente, bzw. von Java-Methodenaufrufen in SOAP-Requests wird die JAX-RPC-Technologie im JBossWS verwendet.

2.3.4 JBossWS und WS-Security

JBossWS verwendet interne *Handlerchains*, zur Verarbeitung von SOAP-Nachrichten. Um WS-Security zur Verschlüsselung von Anfragen zu verwenden, müssen Klient (oder auch Server) den entsprechenden Handler konfigurieren.



Information 35: JBossWS and WS-Security

Information 35 zeigt, dass eine SOAP-Nachricht durch einen Sicherheitsheader gesichert wird. Der Sicherheitsheader enthält Sicherheitstoken, digitale Signaturen und Geheimschlüssel zur Entschlüsselung. Die hierfür geeignete JBossWS-Komponente ist der WSS-Handler. Als Interceptor fängt er ein- und ausgehende Nachrichten ab und verarbeitet ein konkretes WS-Security-Protokoll.

(1) Keystore types
  (1) Java Keystore (jks)
  (2) PKCS12

(2) XML-Signature
  (1) RSA-SHA 1

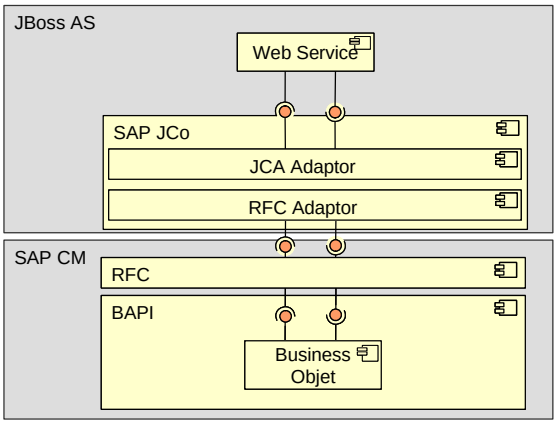
(3) XML Encryption
  (1) 3DES
  (2) AES-128
  (3) AES-196
  (4) AES-256

Information 36: Signature and Encryption with JBossWS

JBossWS unterstützt die Verwendung verschiedener Keystores zur Verschlüsselung und Signierung von SOAP-Nachrichten, diese müssen allerdings auf X.509-Zertifikaten basieren. Zum Erstellen einer Signatur wird der RSA-Algorithmus auf ein SHA-1 Hash angewendet. Zur Verschlüsselung steht neben den gebräuchlichen Verfahren auch das weniger gebräuchliche AES-196 zur Verfügung. Der Transport von Benutzertoken ist auf das *Username Token* beschränkt.

2.3.5 Ankopplung von SAP CM

Information 37 zeigt, wie ein JBoss-AS mit einem SAP CM System über Webservices gekoppelt ist.



Information 37: Interaction JBoss AS and SAP CM

SAP bietet die *Java Connector* (JCo)-Bibliothek an, um auf Funktionsbausteine über das verfügbare RFC-Protokoll (*Remote Function Call*) zuzugreifen und Methoden für beide Kommunikationsrichtungen zur Verfügung zu stellen. Außerdem werden Transaktionen in JCo unterstützt. Die Verwendung von IDOCs (*Intermediary Documents*) ist auch möglich. Alle Klassen und Interfaces der JCo-Bibliothek sind als innere Klasse der Klasse JCo realisiert. Die JCA-Implementierung von SAP stellt eine weitere Abstraktion der JCo-Schnittstelle dar, deren sie sich intern bedient. Mit ihrer JCA-Implementierung gewährleistet SAP eine standardisierte Integration des JCo im J2EE Application Server. Damit kann die JCo-Bibliothek in JBoss-AS integriert werden, um die Kommunikation über Webservices zwischen JBoss-AS und SAP CM aufzubauen.

2.4 Übungsaufgaben

Hinweise zur Lösung der Übungsaufgaben befinden sich am Ende des Dokumentes im Kapitel Lösungshinweise zu den Aufgaben.

?

(1) Which componentes are required to run a BPEL process? Which components are additionally required to handle WS-Security?

(2) Explain differences between gateways and agents!

(3) Considering JBoss AS's Architecture, which components are required to provide and manage web services?

(4) Which components are additionally require to handle WS-Security?

Information 38: DESIGN - Exercises

(1) Welche Komponenten werden zur Ausführung eines BPEL-Prozesses benötigt? Welche Komponenten werden zur Behandlung von WS-Security zusätzlich benötigt? Siehe hierzu Information 18, vergleiche auch mit Information 8.

(2) Erläutere Unterschiede zwischen *Gateways* und *Agents*!

(3) Betrachte die Architektur des JBoss AS, welche Komponenten werden zur Behandlung von Webservices benötigt? Siehe hierzu Information 29 und Information 34.

(4) Welche Komponenten werden zur Behandlung von WS-Security zusätzlich benötigt? Siehe hierzu Information 35.

3 IMPLEMENTIERUNG

Dieses Kapitel beschreibt die Details der Umsetzung von der Umsiedlung der Basis-Webservices über die Installation und Konfiguration der nichtfunktionalen Komponenten zur Absicherung der Webservice-Kommunikation mittels WS-Security bis zur Erweiterung der funktionalen Komponenten um Zugriffskontrolle.

(1) Oracle SOA Suite 10.1.3.10
  (1) Oracle Application Server with OC4J
  (2) Oracle BPEL Process Manager
  (3) Oracle Web Services Manager

(2) JBoss Enterprise Middleware Suite
  (1) JBoss Application Server 4.0.5 with Tomcat 5.5
  (2) JBossWS 1.0.4

(3) Legacy Systems
  (1) SAP Campus Management
  (2) MySQL Databases

Information 39: Systems Overview

Bei der Umsetzung wurden folgende Systeme verwendet:

(1) Die Oracle SOA Suite enthält alle benötigten Komponenten zum Betrieb und zur Absicherung des BPEL-Prozesses.

(2) Die JBoss Enterprise Middleware Suite enthält den Application Server mit Web Container (Tomcat 5.5) und Webservice-Modul JBossWS.

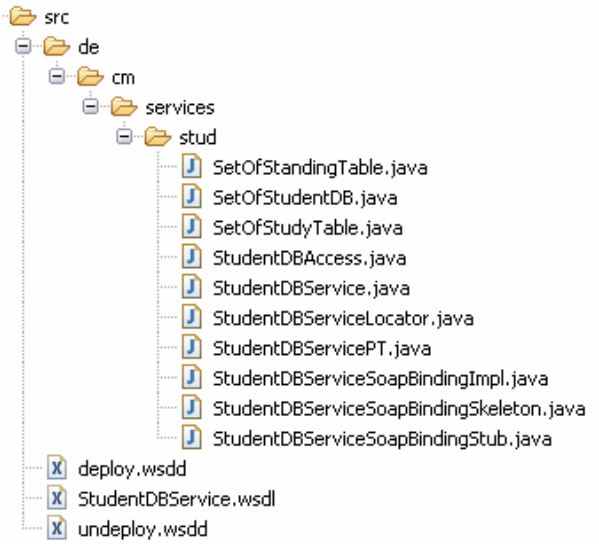
(3) Die vorhandenen Altsysteme wurden unverändert übernommen.

3.1 Migration der Basis-Webservices von AXIS auf den JBoss Application Server

Bei der Umsiedelung der Webservices von AXIS auf den JBoss Application Server soll die bereits vorhandene fachliche Funktionalität und in Verbindung damit auch die Darstellung der Geschäftsobjekte identifiziert und weiterverwendet werden. Zu beachten ist, dass die Entwicklung der AXIS-Webservices in *top down*-Manier geschah, also ausgehend von einer WSDL-Beschreibung wurden mit Hilfe eines Tools sämtliche Klassen, Hilfsklassen, Interfaces und Deployment-Deskriptoren erzeugt. Die bevorzugte Entwicklungspraxis auf dem JBoss AS verläuft *bottom up*, also ausgehend von Java-Klassen, welche Geschäftsfunktionalität und –objekte kapseln werden benötigte Webservice-Artefakte, unter anderen die WSDL-Beschreibung, mit Hilfe von Tools generiert.

3.1.1 Extraktion der Geschäftsfunktionalität aus den alten AXIS-Webservices

Da AXIS-Komponenten intern eine Vielzahl proprietärer Mechanismen nutzen, müssen zunächst diejenigen Abschnitte des bestehenden Codes (Klassen, Interfaces, Webservice-Artefakte) bestimmt, welche die eigentliche Geschäftsfunktionalität implementieren und weiterhin verwendet werden können.



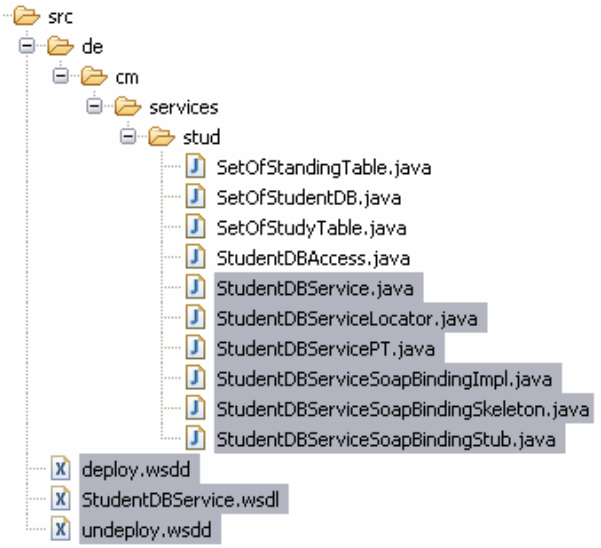
Information 40: Existing StudentDBService

Information 40 zeigt beispielhaft für alle Basis-Webservices die verschiedenen Artefakte der AXIS-spezifischen Implementierung. Die Darstellung stammt aus dem Package-Explorer der Eclipse Entwicklungsumgebung.

(1) Die verschiedenen Artefakte sind innerhalb eines Pakets (hier „de.cm.tm.services.stud“) gruppiert.

(2) Neben Klassen und Interfaces mit der Endung JAVA enthält ein solches Webservice-Paket auch einige XML-Deskriptoren mit der Endung WSDD und die WSDL-Beschreibung mit Endung WSDL.

Zur Bestimmung des AXIS-spezifischen Programmcodes betrachtet man das Tool WSDL2Java, welches bei der Entwicklung von AXIS-Webservices eingesetzt wird, um benötigte Hilfsklassen automatisch zu generieren.



Information 41: AXIS Proprietary Artifacts

(1) Die in Information 41 markierten von WSDL2Java generierten Hilfs-Artefakte können entfernt werden.

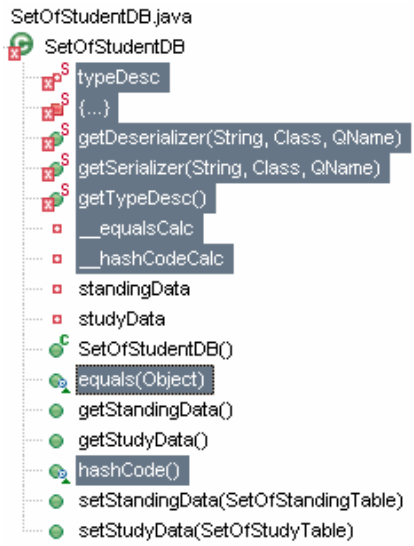
(1.1) AXIS-spezifische Deployment-Deskriptoren bzw. WSDD-Bindings werden nicht weiter benötigt.

(1.2) Klassen und Interfaces, welche sich auf das SOAP-Binding beziehen („SoapBinding“), hierzu gehören auch das „Service“ Interface und der „ServiceLocator“ sind ebenfalls AXIS-spezifisch und werden nicht mehr benötigt.

(1.3) Innerhalb der vorhandenen WSDL-Beschreibungen wird auf das vom WS-I Basic Profile 1.0 abgeschaffte SOAP-Codierungsformat „encoded“ zurückgegriffen. Die alten WSDL-Beschreibungen werden wie auch deren Java-Entsprechungen (bei AXIS: PortType-Interface „PT.java“) nicht weiter benötigt.

(2) Die restlichen Artefakte kapseln Geschäftsfunktionalität („DBAccess.java“) und Geschäftsobjektsklassen („SetOf…..java“).

Bei näherer Betrachtung enthalten auch diese Fragmente AXIS-bezogenen Code, da sie von WSDL2Java aus komplexen Typen, die innerhalb der WSDLs definiert werden, erzeugt wurden. Auch dieser AXIS-bezogene Code sollte entfernt werden.

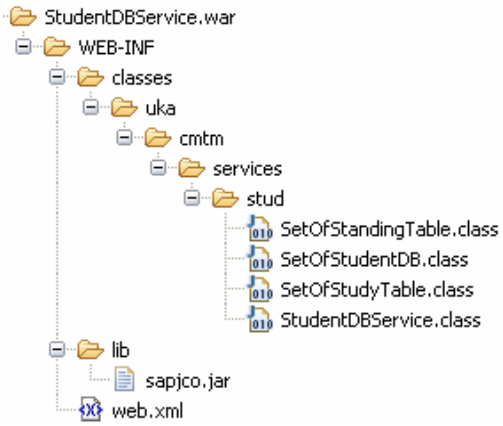


Information 42: AXIS Proprietary Code Fragments

Information 42 zeigt am Beispiel der Geschäftsobjektsklasse „SetOfStudentDB“ welche Code-Abschnitte AXIS-spezifisch sind und damit entfernt werden sollten (Markierung) und welche Abschnitte weiterhin Verwendung finden (keine Markierung). Vergleiche hierzu auch mit Information 10.

3.1.2 Entwicklung der neuen JBoss-Webservices

Die ToR-Webservices werden bei dieser Fallstudie als JSR-181 Endpunkte für Plain Old Java Objects (POJO) realisiert, da dies die einfachste und schnellste Methode ist, um Webservices mit dem JBoss AS zu entwickeln. Am Beispiel des „StudentDBServices“ wird ein generisches Vorgehen zur Entwicklung der Webservice-Endpunkte gezeigt.



Information 43: StudentDBService JSR-181 POJO Endpoint

Die gezeigte Verzeichnisstruktur ist typisch für so genannte POJO-Komponenten.

(1) Das zentrale Verzeichnis „WEB-INF“ enthält alle zur Komponente gehörenden Artefakte.

(2) Das Unterverzeichnis „classes“ enthält Klassen und Interfaces. Die untergeordnete Verzeichnisstruktur ist hierbei aus dem ursprünglichen Paketnamen abgeleitet. (Hinweis: „StudentDBService“ entspricht hier der alten Klasse „StudentDBAccess“, außerdem wurde der Paketname abgeändert. Vergleiche mit Information 41).

(3) Das Unterverzeichnis „lib“ enthält externe Bibliotheken, die zur Ausführung der Komponente benötigt werden. Hier handelt es sich um die SAP JCo Laufzeit-Bibliothek „sapjco.jar“, für den Zugriff auf MySQL-Datenbanken wird ein entsprechender JDBC-Treiber benötigt.

(4) Der Deployment-Deskriptor „web.xml“ enthält Informationen, welche ein Web-Container benötigt um diese Komponente auszuführen und zu verwalten. Siehe hierzu Information 44.

Kapitel 3.1.1 zeigt exemplarisch wie aus den alten AXIS-Webservices die benötigte Geschäftsfunktionalität extrahiert wird. Um daraus nun Webservices zu erhalten, die auf dem JBoss AS betrieben werden können, muss noch der Deployment-Deskriptor „web.xml“ angepasst werden. Zur Erstellung von JSR-181 POJO-Webservices siehe [C&M-Tech-JBoss].

```
<?xml version=“1.0” encoding=“UTF-8”?>
<web-app version=“2.4” xmlns=“http://java.sun.com/xml/ns/j2ee”>
  <servlet>
    <servlet-name>StudentDBService</servlet-name>
    <servlet-class>
      uka.cmtm.services.stud.StudentDBService
    </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>StudentDBService</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Information 44: Listing 1 - web.xml

(1) Zum Einen gibt der Deskriptor „web.xml“ der Komponente einen Namen <servlet-name>, bei dieser POJO-Komponente handelt es sich streng genommen also um ein Servlet.

(2) Zum Anderen, was wesentlich wichtiger ist, zeigt der Deskriptor auf die eigentliche Geschäftsklasse <servlet-class>.

Zum Deployment der Webkomponente: kopiere das erstellte WAR-Paket in das Verzeichnis „%JBOSS-HOME%\server\default\deploy“. Danach werden die Webservices aufgelistet unter folgender Adresse http://localhost:8080/jbossws/services. Die WSDL-Beschreibung wird unter http://localhost:8080/StudentDBService/StudentDBService?wsdl veröffentlicht.Wiederhole die obigen Schritte, um die anderen POJO-Komponenten (ExaminationDB, CourseUnitDB und OrganizationDBService) zu erstellen. Die vier POJO-Komponenten mit alten Funktionalitäten werden mit Hilfe des Deployment-Deskriptors „web.xml“ in den Web Container von JBoss AS betrieben.

3.2 Absicherung der Webservice-Aufrufe mit dem Oracle Web Services Manager

Zur Signierung und Verschlüsselung der Webservice-Kommunikation sind auf Oracle-Seite folgende Konfigurationen vorzunehmen:

(1) X.509 certificates
  (1) Generate certificates
  (2) Distribute certificates to keystores

(2) OWSM gateway
  (1) Install gateway
  (2) Register web services
  (3) Configure Security Policies

(3) BPEL process
  (1) Invoke the secured gateway web services

Information 45: Requirements

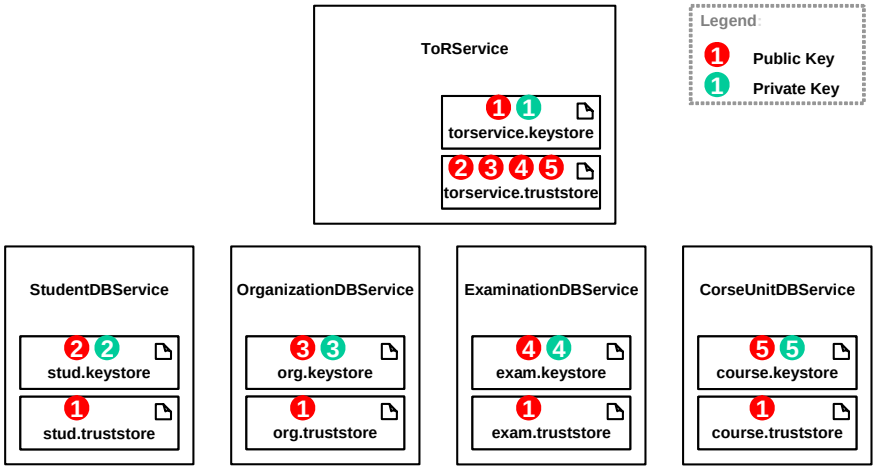
(1) Mit dem Java keytool lassen sich X.509 Zertifikate erstellen. Man kann dabei verschiedene Optionen wählen, z.B. um ein bestimmtes Schlüsselformat zu generieren. Hier wurde beispielsweise „-keyalg RSA“ gewählt, um den Schlüssel zur Signierung mit „RSA SHA 1“ und bei der Verschlüsselung zum Austausch der symmetrischen Schlüssel mit „RSA 1.5“ zu verwenden.

(2) Die Installation und Konfiguration eines OWSM-Gateways kann vollständig über die OWSM-Konsole geschehen. Erreichbar ist diese Konsole z.B. über die Konsole „Application Server Control“, welche sich üblicherweise unter der Adresse „http://localhost:8888“ befindet. Eine detaillierte Anleitung gibt der nächste Abschnitt.

(3) Schließlich werden vom BPEL-Prozess aus die Basis-Webservices nicht mehr direkt aufgerufen, sondern die abgesicherten virtuellen Endpunkte, die vom OWSM-Gateway bereitgestellt werden. Hierzu müssen die Adressen der jeweiligen PartnerLinks entsprechend angepasst werden. Die Adressen der Gateway-Webservices können über die OWSM-Konsole aufgelistet werden.

3.2.1 Schlüsselverteilung

Für jede beteiligte Partei wird mit dem „Java keytool“ ein Schlüsselpaar generiert und anschließend die öffentlichen Schlüssel so verteilt, dass jeder benötigte Kommunikationsweg abgesichert werden kann.

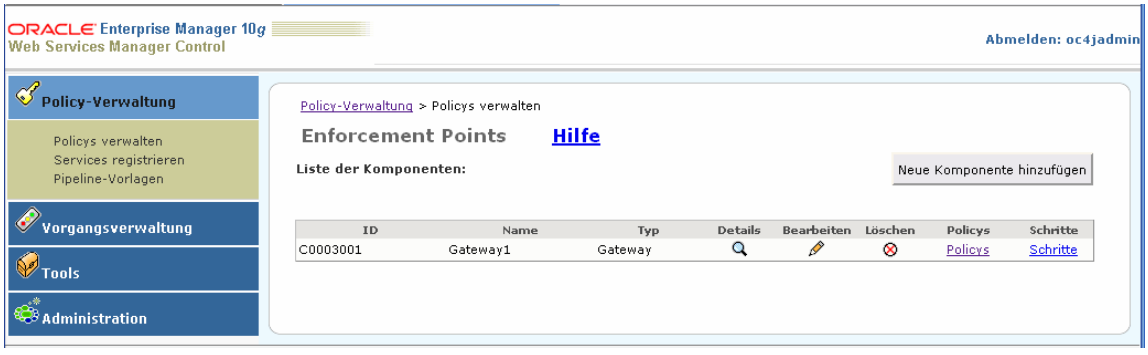


Information 46: Key Distribution

Sowohl der OWSM als auch JBossWS unterstützen für diesen Zweck zwei getrennte Keystores: einen Keystore, der das eigene Schlüsselpaar enthält und einen Truststore, der die öffentlichen Schlüssel der vertrauenswürdigen Kommunikationspartner enthält. Im Truststore, der dem BPEL-Prozess zugeordnet ist, befinden sich daher die öffentlichen Schlüssel aller vier Basis-Webservices, in den Truststores der Basis-Webservices wird lediglich der öffentliche Schlüssel des BPEL-Prozesses benötigt.

3.2.2 Oracle Web Services Manager

Im Folgenden wird beschrieben, wie ein OWSM-Gateway installiert, Webservices registriert und Policies über die OWSM-Konsole konfiguriert werden. Über den Menüpunkt „Policy-Verwaltung“ werden alle hierfür benötigten Funktionen angeboten.



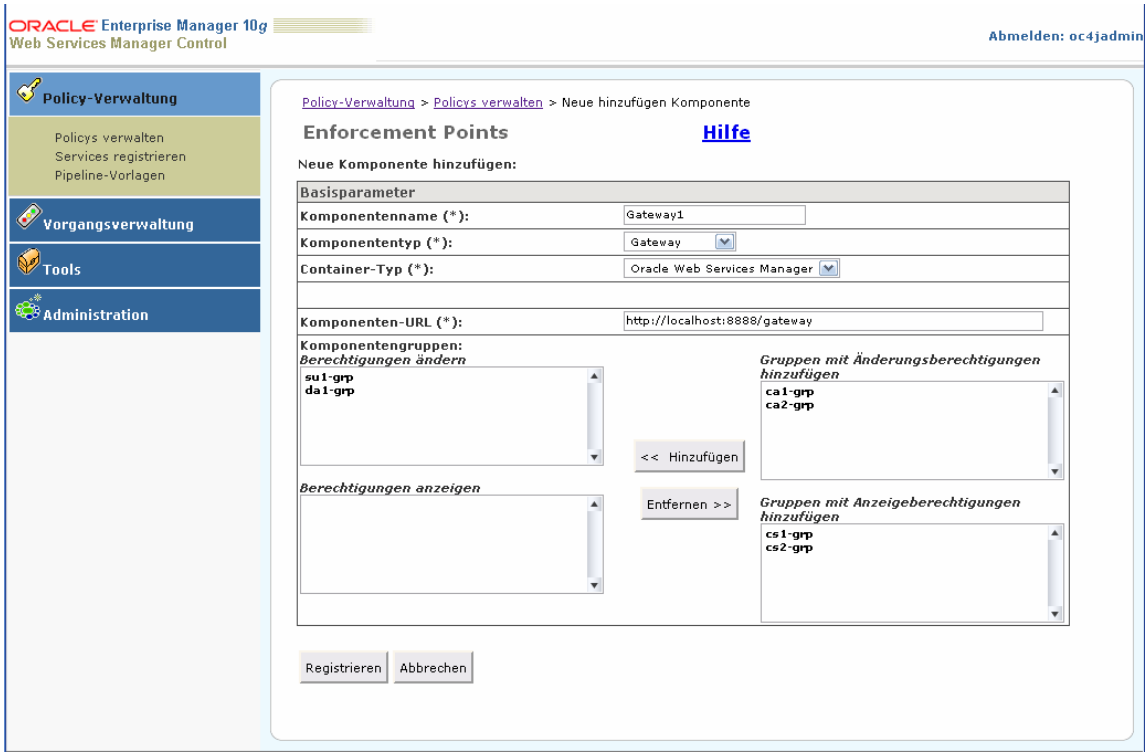
Information 47: Oracle Web Services Manager Control Console

(1) Über „Policys verwalten“ können OWSM-Komponenten, also Gateways oder Agents, hinzugefügt und konfiguriert werden. Information 47 zeigt diese Seite nachdem bereits ein Gateway installiert wurde. Jede Komponente erhält eine eindeutige Kennung („ID“) zugewiesen und verfügt zur besseren Bedienbarkeit über einen frei wählbaren Namen. Nach der Installation einer Komponente können „Policys“ angepasst und hierfür verfügbare „Schritte“ eingesehen werden.

(2) Über den Menüpunkt „Services registrieren“ können für verschiedene Gateways Webservices registriert werden. Dabei wird ein virtueller Webservice-Endpunkt erstellt, der den ursprünglichen Webservice verschattet. So kann die Kommunikationsstrecke zwischen ursprünglichem Webservice und Gateway abgesichert werden.

(3) Über den Menüpunkt „Pipeline-Vorlagen“ können Vorlagen erstellt werden, die dann innerhalb konkreter Policies als eine Reihe von Policy-Schritten eingefügt werden können.

Über „Policy-Verwaltung“ > „Policys verwalten“ > „Neue Komponente hinzufügen“ wird ein Gateway installiert.



Information 48: OWSM Control Console - Add an OWSM Gateway Component

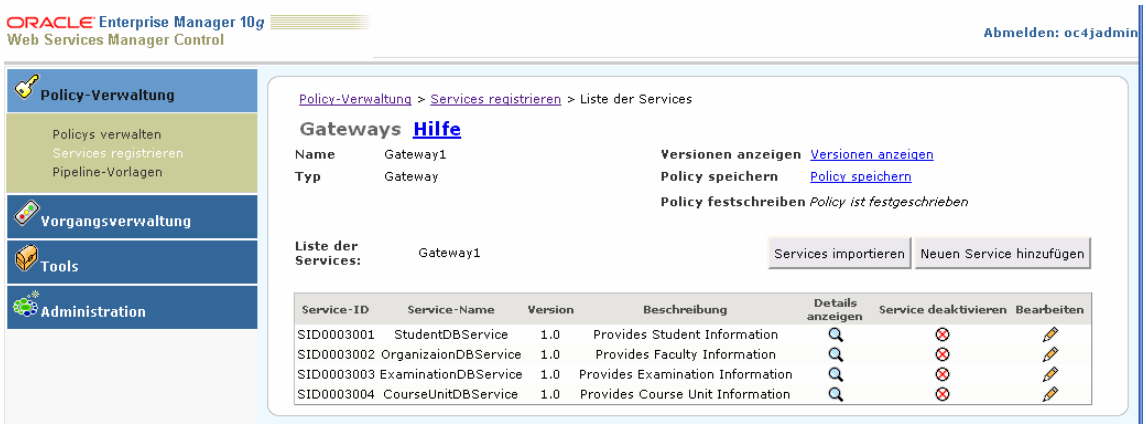
(1) Der Komponentenname kann frei gewählt werden, ein Vorschlag wäre “Gateway1”.

(2) Als Komponententyp muss „Gateway“ gewählt werden, da zur Absicherung des BPEL-Prozesses keine *Agents* eigesetzt werden können.

(3) Der Container-Typ „Oracle Web Services Manager“ ist fest vorgegeben.

(4) Schließlich wird die Komponenten-URL spezifiziert, unter welcher das Gateway später zu erreichen ist. Das Adress-Schema ist auch hier vorgegeben. Es handelt sich hierbei um die HTTP-Basisadresse des Servers „http://localhost:8888/“ erweitert um den Adressanteil des Gateways „gateway“. Die Installation des Gateways wird über die Schaltfläche „Registrieren“ abgeschlossen.

Als nächstes werden die Basis-Webservices am Gateway registriert.



Information 49: OWSM Control Console - Register Services

Über „Policy-Verwaltung“ > „Services registrieren“ wird eine Reihe installierter Gateways aufgelistet. Hier werden am entsprechenden Gateway „Services“ angemeldet.

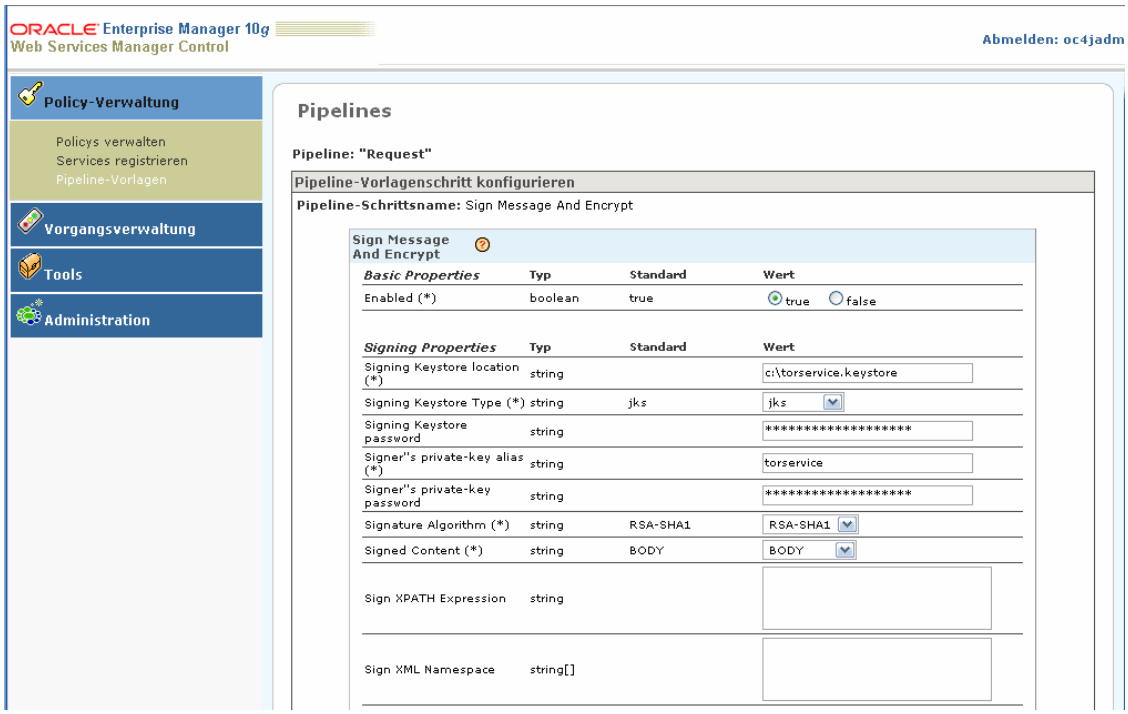
(1) Über „Neuen Service hinzufügen“ werden die einzelnen Basis-Webservices am Gateway registriert. „Service-Name“ und „Service-Version“ sind frei wählbar. Unter „WSDL URL“ wird die WSDL-Adresse des jeweiligen Webservices angegeben. „Service-Protokoll“ gibt das genutzte Transportprotokoll an, also „HTTP(S)“ für HTTP oder HTTP/S. Auf der nächsten Seite werden diese Angaben nochmals bestätigt.

(2) Nachdem alle Basis-Webservices registriert wurden, kann die „Liste der Services“ wie in Information 49 unter „Policy-Verwaltung“ > „Services registrieren eingesehen werden.“ Über den Punkt „Bearbeiten“ > „--> Policy ändern“ kann für jeden Webservice separat eine Sicherheits-Policy angelegt werden. Alternativ kann zunächst eine Pipeline-Vorlage erstellt werden.

(3) „Policy speichern“ schreibt den kompletten Satz an Policies des ausgewählten Gateways in eine XML-Datei „PolicySet.xml“.

(4) Sobald für einen der Webservices Änderungen an dessen Policy vorgenommen wurden, wird die Schaltfläche „Policy festschreiben“ aktiviert. Auf diese Weise werden die angelegten Policies einem Gateway bekanntgemacht. Beim nächsten Aufruf eines registrierten Webservices versucht das Gateway diese Policies durchzusetzen.

Über „Policy-Verwaltung“ > „Pipeline-Vorlagen“ können neue Pipeline-Vorlagen erstellt werden. Hierzu zunächst Komponententyp „Gateway“ und Pipeline-Typ „Request“ wählen, dann „Neue Pipeline-Vorlage hinzufügen“. Als Name „InvokeSecuredServiceRequest“ wählen, „Weiter“. Auf der nächsten Seite wird über „Schritt unten hinzufügen“ der Policy-Schritt „Sign Message and Encrypt“ hinzugefügt.

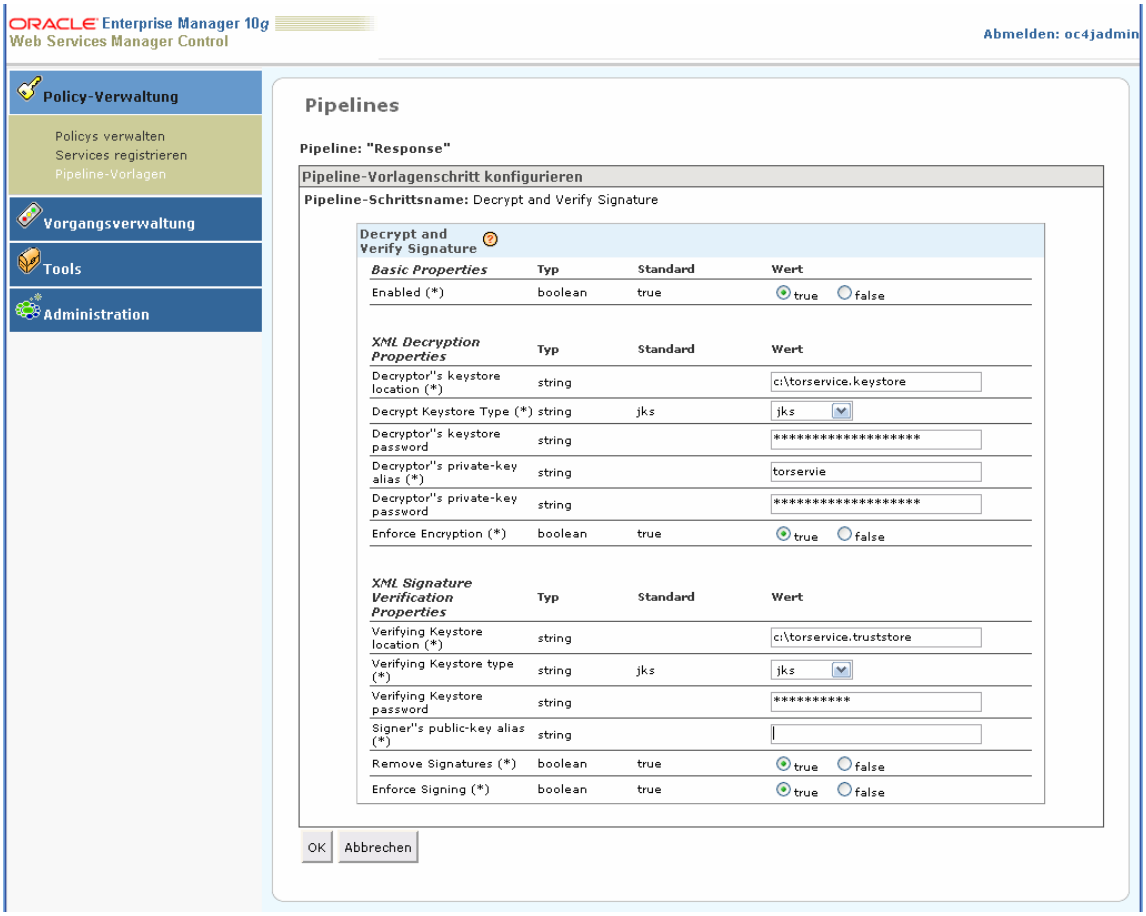


Information 50: OWSM Control Console - Pipeline Templates 1

(1) „Signing Keystore location“ zeigt auf den Speicherort des Keystores, z.B. „c:\torservice.keystore“. Weiter müssen das Passwort für den Keystore, das Alias und das Passwort für den privaten Schlüssel angegeben werden. Der Algorithmus zur Signierung wird auf „RSA-SHA1“ gesetzt, der zu signierende Inhalt ist der Nachrichtenrumpf, also der SOAP Body.

(2) In ähnlicher Weise werden die „Encryption Properties“ konfiguriert. Zur Verschlüsselung wird der öffentliche Schlüssel des Empfängers verwendet. „Decrypter’s public-key alias“ muss also später nochmals für jeden Service einzeln angepasst werden. Die „Encryption Keystore location“ zeigt auf den Truststore, z.B. „c:\torservice.truststore“. Verschlüsselt wird mit „AES-128“, die hierfür benötigten symmetrischen Schlüssel werden mit „RSA-1.5“ ausgetauscht.

Auch die Behandlung der Antwort (Pipeline-Typ „Response“) kann in einer Pipeline-Vorlage in ähnlicher Weise vorkonfiguriert werden. Als Name kann z.B. „InvokeSecuredServiceResponse“ gewählt werden. Als Pipeline-Schritt wird dann „Decrypt and Verify Signature“ hinzugefügt.



Information 51: OWSM Control Console - Pipeline Templates 2

(1) Unter "XML Decryption Properties werden Angaben zur entschlüsselung gemacht. Da der symmetrische Schlüssel mit dem öffentlichen Schlüssel des BPEL-Prozesses verschlüsselt wurde, wird zur Entschlüsselung der private Schlüssel des BPEL-Prozesses verwendet.

(2) Zur Verifizierung der Signatur wird der öffentliche Schlüssel des jeweiligen Services benötigt, sodass dieser Eintrag später für jeden Basis-Webservice nochmals angepasst werden muss.

Diese beiden Pipeline-Vorlagen können als Sicherheits-Poilies zur Absicherung der Kommunikation mit den Basis-Webservices verwendet werden. Über „Policy-Verwaltung" > „Services registrieren" > „Services" kann unter dem Punkt „Bearbeiten" für jeden einzelnen Webservice die jeweilige Sicherheitspolicy angepasst werden („-->Policy ändern"). Hierzu wird die jeweils vorhandene Pipeline-Vorlage mit den eben erstellten ersetzt („Pipeline ersetzen"). In diesem Zuge werden die beiden genannten Punkte, der Schlüssel-Alias bei der Verschlüsselung einer Anfrage und der Schlüssel-Alias bei Verifizierung der Signatur, angepasst. Abschließend werden die Policies wie weiter oben erklärt an das Gateway weitergereicht („Policy festschreiben") und die PartnerLinks im BPEL-Prozess angepasst.

3.3 Absicherung der Basis-Webservices mit dem JBoss Application Server

(1) Servlet deployment descriptor „web.xml“

(2) JBossWS configuration descriptor „jboss-wsse-server.xml“

(3) Create keystores

Information 52: Securing Web Service Endpoints on JBoss AS

Information 52 zeigt, was benötigt wird, um einen abgesicherten Webservice von JBoss-AS zu erstellen.

3.3.1 JBossWS - Konfiguration

Der Servlet-spezifische Deployment-Descriptor „web.xml“ muss wie folgt eingestellt werden, um einen Standard-Sicherheitsendpunkt zu spezifizieren, das heißt den WS-Security-Handler für ein- und ausgehende SOAP-Nachrichten zu aktivieren:

```
<web-app>
  <context-param>
(1)  <param-name>jbossws-config-name</param-name>
(2)  <param-value>Standard Secure Endpoint</param-value>
  </context-param>
  <servlet>
    <servlet-name>StudentDBService</servlet-name>
    <servlet-class>
      uka.cmtm.services.stud.StudentDBService
    </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>StudentDBService</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Information 53: Listing 2 - web.xml

(1) JBossWS spezifische Kontextparameter.

(2) Zuweisen zu „Standard Secure Endpoint“.

Die Konfiguration der WS-Security-Impelementierung geschieht auf Server-Seite (Service Provider) über den JBoss-spezifischen Deployment-Descriptor „jboss-wsse-server.xml“. Auf dem Server müssen spezifizierte Key-Store, Trust-Store und ein konkretes WS-Security-Protokoll konfiguriert werden.

```
<jboss-ws-security>
(1)<key-store-file>WEB-INF/stud.keystore</key-store-file>
(2)<key-store-password>jbossws</key-store-password>
(3)<trust-store-file>WEB-INF/stud.truststore</trust-store-file>
(4)<trust-store-password>jbossws</trust-store-password>
(5)<config>
(6)   <sign type="x509v3" alias=„stud" />
       <encrypt type="x509v3" alias=„tor" />
(7)    <requires>
          <encryption />
(8)       <signature />
       </requires>
    </config>
</jboss-ws-security>
```

Information 54: Listing 3 - jboss-wsse-server.xml

(1) Hinweise zur Key-Store-Datei, die sich in der WAR-Datei unter dem Namen „WEB-INF/stud.keystore“ befindet.

(2) Ein Passwort mit dem Wert „jbossws“.

(3) Hinweise zur Trust-Store-Datei, die sich in der WAR-Datei unter dem Namen „WEB-INF/stud.truststore“ befindet.

(4) Ein Passwort mit dem Wert „jbossws“.

(5) Der Konfigurationsblock <config> enthält Default-Konfigurationen für alle Services, die in der WAR-Datei verwendet werden.

(6) Der Server muss den Nachrichtenrumpf aller Antworten signieren. Das Attribut „type“ bezeichnet das Kodierungsformat des Schlüssels, der Standardwert „X509v3“ ist auch das einzig zulässige. Das Attribut „alias“ gibt an, welcher Zertifikat-Alias im Key-Store, und damit welcher Schlüssel, zur Signierung benutzt wird.

(7) Der Requiresblock <requires> spezifiziert alle Sicherheitsanforderungen, die der Server beim Empfangen einer Nachricht treffen muss.

(8) Dieses Element hat die Bedeutung, dass alle Webservices in dieser WAR-Datei aufgefordert werden, den Nachrichten-Body zu signieren.

3.3.2 Schlüsselerzeugung

Das „Java Keytool“-Kommando zur Herstellung eines Schlüsselpaares für den Key-Store lautet folgendermaßen:

```
keytool -genkey -alias stud -keyalg RSA -keysize 2048 -keystore stud.keystore
-dname “CN=Y. Zhao S. Kreuz, OU=cmtm, O=Universitaet Karlsruhe, C=de”
```

Information 55: Create Keystore

(1) Gebe Passwort für Key-Store ein: jbossws

(2) Gebe Passwort für Schlüssel ein: jbossws

Das „Java Keytool“-Kommando zur Herstellung eines Schlüsselpaares für den Trust-Store lautet folgendermaßen:

```
keytool –genkey –alias stud –keyalg RSA –keysize 2048 -keystore stud.truststore
-dname “CN=Y. Zhao S.Kreuz, OU=cmtm, O=Universitaet Karlsruhe, C=de”
```

Information 56: Create Truststore

(1) Gebe Passwort für Trust-Store ein: jbossws

(2) Gebe Passwort für Schlüssel ein: jbossws

Austausch der SOAP-Nachricht

Hier folgt ein Beipiel einer SOAP-Nachricht (siehe Information 57):

```
<env :Envelope
    xmlns :env=“http://schemas.xmlsoap.org/soap/envelope/“>
  <env:Header>
    <wsse:Security env:mustUnderstand=”1” …>
      <wsu:Timestamp wsu:Id=”timestamp”>…</wsu:Timestamp>
      <wsse:BinarySecurityToken …>
      […]
      </wsse:BinarySecurityToken>
      <ds:Signature xmlns:ds=”http://www.w3.org/2000/09/xmldsig#”>
      […]
      </ds:Signature>
    </wsse:Security>
  </env:Header>
  <env:Body wsu:Id=”element-1-1140197309843-12388840” …>
    […]
  </env:Body>
</env:Envelope>
```

Information 57: SOAP Message

Der SOAP-Body bleibt in der Plain-Text Form. Aber die SOAP-Nachricht wird im Sicherheits-Header signiert, damit sie bei Übertragung nicht manipuliert werden kann.

3.4 Übungsaufgaben

Hinweise zur Lösung der Übungsaufgaben befinden sich am Ende des Dokumentes im Kapitel Lösungshinweise zu den Aufgaben.

?

(1) Where are the benefits of migrating the Web Services from AXIS to JBoss AS?

(2) How and why are keys distributed? Which keys are required to encrypt messages?

(3) What happens, if another basic web service is added (if another BPEL process needs access)?

Information 58: IMPLEMENTATION - Exercises

(1) Welche Vorteile können von der Migration der AXIS-Webservices auf den JBoss AS erwartet werden? Vergleiche hierzu auch Information 40 mit Information 43.

(2) Wie und warum werden Schlüssel verteilt? Welche Schlüssel werden zur Verschlüsselung von Nachrichten verwendet?

(3) Wie ändert sich die Schlüsselverteilung wenn ein weiterer Basis-Webservice hinzukommt (ein weiterer BPEL-Prozess die Dienste nutzen möchte)?

4 AUSBLICK

Die Absicherung von Webservice-Interaktionen ist eine zwingend notwendige Vorraussetzung für den Einsatz dieser Technologien im Unternehmenssoftwarebereich. Die vorliegende Fallstudie hat gezeigt welche Anforderungen aus geschäftlicher und technischer Sicht bestehen, welche Infrastruktur-Komponenten zusätzlich benötigt werden und welche weiteren Vorbedingungen hinsichtlich der Absicherung bestehen. Dabei ist die Absicherung von Information während der Übertragung nur ein Teilschritt in Richtung sicherer Softwaredienste, ein weiterer Schritt, Zugriffskontrolle, wurde angesprochen. Beide Absicherungsmaßnahmen beziehen sich hier ausschließlich auf den *Application Layer* (Information 14). Bei der Durchführung dieser Fallstudie konnten verschiedene Probleme und Mögliche Lösungsansätze ausgemacht werden. Diese werden im Folgenden diskutiert.

4.1 Interoperabilität

Aufgrund bestehender Interoperabilitätsprobleme zwischen den WS-Security-Handlern des Oracle Web Services Managers und des JBoss Application Servers wurden abgesicherte Nachrichten mit Fehlermeldungen abgelehnt. Information 59 zeigt einen Ausschnitt aus einer abgesicherten Anfrage des „Transcript of Records“-Services an den „StudentDBService“.

```
(24)          <dsig:Signature>
(25)             <dsig:SignedInfo>
(26)                <dsig:CanonicalizationMethod Algorithm='xml-exc-c14n#' />
(27)                <dsig:SignatureMethod Algorithm='#rsa-sha1' />
(28)                <dsig:Reference URI='#upvv0oKRns01bbB8K9gt0g22'>
(29)                   <dsig:Transforms>
(30)                      <dsig:Transform Algorithm='xml-exc-c14n#' />
(31)                   </dsig:Transforms>
(32)                   <dsig:DigestMethod Algorithm='#sha1' />
(33)                   <dsig:DigestValue>
(34)                      L2yCxnF/RUn6/mKrogixZRxXc04=
(35)                   </dsig:DigestValue>
(36)                </dsig:Reference>
(37)                <dsig:Reference URI='#4LH1bFdEHeSu7ZSikGjn9g22'>
(38)                   <dsig:Transforms>
(39)                      <dsig:Transform Algorithm='xml-exc-c14n#' />
(40)                   </dsig:Transforms>
(41)                   <dsig:DigestMethod Algorithm='#sha1' />
(42)                   <dsig:DigestValue>fPkwRC/uMv4dImCvH+8zqhLNkfs=</dsig:DigestValue>
(43)                </dsig:Reference>
(44)             </dsig:SignedInfo>
(45)             <dsig:SignatureValue>PWJ8hJ4JCR3fFz2j0uyICHz9Yj/8iajI...</dsig:SignatureValue>
(46)             <dsig:KeyInfo>
(47)                <SecurityTokenReference wsu:Id='_3mzvf1VVEi7eWSTyKk8HCg22'>
(48)                   <Reference URI='#_bC6KBvAOzAzDT8cPzTYI2g22' />
(49)                </SecurityTokenReference>
(50)             </dsig:KeyInfo>
(51)          </dsig:Signature>
(52)          <wsu:Timestamp wsu:Id='4LH1bFdEHeSu7ZSikGjn9g22'>
(53)             <wsu:Created>2007-01-18T22:47:21Z</wsu:Created>
(54)          </wsu:Timestamp>
(55)       </wsse:Security>
(56)    </soap:Header>
(57)    <soap:Body wsu:Id='upvv0oKRns01bbB8K9gt0g22'>
(58)       <xenc:EncryptedData Id='_OmxcuNXdl4R2vtGGXUe0qg22' Type='#Content'>
(59)          <xenc:EncryptionMethod Algorithm='#aes128-cbc' />
(60)          <xenc:CipherData>
(61)             <xenc:CipherValue>Q7kaIvnfyVQPJ/Lie/fGCJN8InU1S6hT...</xenc:CipherValue>
(62)          </xenc:CipherData>
(63)       </xenc:EncryptedData>
(64)    </soap:Body>
(65) </soap:Envelope>
```

OASIS WS-Security 1.0

Line 708:
The usage of **ValueType** is RECOMMENDED for references with local URIs.

WS-I Basic Security Profile 1.0

R3059:
Any **<Reference/>** MUST specify a **ValueType** attribute.

Information 59: Interoperability Issues – Exchanged Message

(1) Der Oracle Web Services Manager setzt an der rot markierten Stelle den OASIS-Standard WS-Security 1.0 vergleichsweise großzügig um, welcher das Vorkommnis des Attributs “ValueType” innerhalb eines <wsse:Reference>-Elementes lediglich empfiehlt.

(2) Der JBoss Application Server hält sich bei seiner Implementierung bereits an den Entwurf des zukünftigen Interoperabilitätsstandards WS-I Basic Security Profiles 1.0, welches das Attribut „ValueType“ hier zwingend vorschreibt. In Folge wird die ansonsten standardkonforme Nachricht als fehlerhaft abgewiesen.

Solche Interoperabilitätsprobleme werden aller Vorraussicht nach mit der Zeit durch die Arbeiten der WS-I gelöst. Durch diese Fallstudie ist deutlich geworden, wie wichtig diese

zusätzliche Arbeit der WS-I für die Einsetzbarkeit von Webservice-Technologien ist. Betrachtet man nun nochmals Information 13, und bedenkt dabei, dass dies auch gerade nur ein Ausschnitt aus der langen Liste der Anforderungen alleine im Bereich Webservice-Sicherheit darstellt und vergegenwärtigt sich wie langwierig und komplex Standardisierungsbemühungen auf der technischen Ebene aber auch auf der Verständnissebene der beteiligten Personen vorangehen, wird deutlich, dass diese Technologien auch ihren Preis haben.

4.2 Schlüsselverteilung

Das Problem der Schlüsselverteilung wurde in dieser Fallstudie händisch angegangen, Mit einem Kommandozeilenwerkzeug wurden Zertifikate erstellt und per *copy and paste* auf die verschiedenen Systeme verteilt. In einem Szenario dieser Größenordnung spricht gegen ein solches Vorgehen auch weiter nichts. Bei der gedanklichen Abbildung in den Wirkbetrieb, tauchen bereits Probleme auf. Während eine zentrale Studentendatenbank sowie eine oder zumindest wenige Datenbanken mit Prüfungsergebnissen durchaus realistisch erscheinen, müssen Fakultätsdatenbanken und Datenbanken mit Vorlesungsinformationen mit einer zweistelligen Anzahl eingeplant werden und genauso viele Administrationsbereiche. Die Zahl der Schlüssel wächst allerdings nur linear, beim hinzufügen einer weiteren Basisdienst-Komponente werden zwei öffentliche Schlüssel ausgetauscht.

An dieser Stelle sei auch nochmals darauf hingewiesen, dass WS-Security in diesem Sinne keine neuartige Sicherheitsinfrastruktur bereitstellt, sondern vorhandene Infrastrukturen integrieren und erweitern will. Zur Verteilung von Schlüsseln und Zertifikaten können bestehende, von WS-Security unabhängige Mechanismen weiterhin genutzt werden, gerade wenn Zertifikate auch zur Authentifizierung eingesetzt werden sollen.

Beim Einrichten einer weiteren Dienstkomposition, welche die hier vorgestellten Basisdienste nutzen können soll gibt es enorme Probleme, wenn der neue Prozess in einen anderen Administrationsbereich fällt, als der ToR-Service, da es nicht denkbar ist zwischen den beiden Prozessbesitzern private Schlüssel auszutauschen. Das Prinzip der losen Kopplung wird von den vorgefundenen WS-Security-Implementierungen ausgeschaltet, für eine Lösung wären also Umwege (z.B. Asynchrone zustandsbehaftete Webservices) oder weitere Technologien (z.B. WS-SecureConversation) notwendig.

4.3 Schnittstellenerweiterung

Ein weiterer Punkt, welchen die Fallstudie deutlich gemacht hat, ist der Mangel einer adäquaten Schnittstellenerweiterung. Gemeint ist, dass WSDL keine sprachlichen Elemente bereitstellt, mit welchen Sicherheitsanforderungen eines Webservices an die Kommunikation und den Zugriff beschrieben werden können. Ein sinnvolles Vorgehen bei der Entwicklung einer Webservice-basierten SOA ist es mit der Beschreibung der Schnittstellen zu beginnen und danach auf Komponentenebene herunter zu steigen, so genannte *contract driven development*. Das hat den Vorteil, dass verschiedene Entwicklerteams unabhängig von einander parallel arbeiten können. Außerdem könnten Schnittstellen auf diese Weise als einheitliche Spezifikation und als einheitliche Konfiguration verwendet werden.

Als Lösungsansatz ist in diesem Bereich WS-Policy zusammen mit WS-PolicyAttachment und WS-SecurityPolicy in Sicht. Daneben stehen allerdings bereits auch weitere Standards, XACML könnte als Framework zur Beschreibung und Durchsetzung von Policies ebenfalls Anwendung finden [An06].

Lösungshinweise zu den Aufgaben

Analyse

(1) Zur Erstellung eines „Transcript of Records“-Notenauszuges werden zunächst verteilte Information gesammelt. Um welche Informationen handelt es sich hierbei?

Lösungshinweis:

Siehe hierzu Information 10: Business Object Model: Transcript of Records.

(2) Vergleiche Information 6 mit dem Prozess in Information 8, wo bestehen Unterschiede? Notiere den Prozess in der Business Process Execution Language (BPEL) unter Verwendung von basic activities und structured activites!

Lösungshinweis:

Information 6 stellt die wesentliche Struktur des ToR-Prozesses in der UML-Notation als Aktivitätsdiagramm dar, derselbe Prozess ist in Information 8 mit der Business Process Modeling Notation (BPMN) dargestellt, allerdings erweitert um die Überprüfung der Matrikelnummer auf Gültigkeit. Zudem deutet Information 8, auf welche Systeme innerhalb des Prozessablaufs zugegriffen wird.

```
<process name=“ToRService“>
<receive name=”RequestTranscriptOfRecords”
partnerLink=”ToRService”>
<invoke name=“LookupStudent“
partnerLink=“StudentDBServicePL“
operation=”isStudent”/>
<switch name=“valid“>
<case condition=” ’validStudent’=’true’ ”>
        <scope name=”validStudentBranch”>
                <flow name=”GetInformation”>
                        <sequence name=”flowSequence1”>
                                <invoke name=”GetStudentInformation”
partnerLink=”StudentDBServicePL”
operation=”getCompleteSet”/>
                                <invoke name=”GetOrganisationInformation”
partnerLink=”OrganisationDBServicePL”
operation=”getCompleteSet”/>
                                </sequence>
                        <sequence name=”flowSequence2”>
                                <invoke name=”GetExaminationInformation”
partnerLink=”ExaminationDBServicePL”
operation=”getCompleteSet”/>
                                <invoke name=”GetCourseUnitInformation”
partnerLink=”CourseUnitDBServicePL”
operation=”getCompleteSet”/>
                        </sequence>
                </flow>
                <sequence name=”CreateTranscriptOfRecords”/>
        </scope>
<otherwise>
```

```
        <scope name="invalidStudentBranch">
                <reply name="ReturnErrorMessage"/>
        </scope>
</switch>
<reply name="ReturnTranscriptOfRecords"/>
</process>
```

(3) Welchen Vorteil bringt die Verwendung von Webservice-Technologien in diesem Szenario? Wo werden hier Webservice-Technologien eingesetzt, gibt es hierfür Alternativen?

Lösungshinweis:

Allgemeine Konzepte hinter Webservices: lose Kopplung, Interoperabilität,…
Zur Verwendung siehe Information 8.
Eine alternative Technologie zur Lösung verteilter Integrationsprobleme ist beispielsweise die Common Object Request Broker Architecture (CORBA).

(4) Der „Transcript of Records"-Service verarbeitet sensible Informationen. Beschreibe grundlegende Sicherheitsanforderungen aus rein geschäftlicher Sicht (aus technischer Sicht)!

Lösungshinweis:

Siehe hierzu Information 7: Threat Scenario: Transcript of Records Service, bzw. Information 14: Securing exchanged Messages within the Transcript of Records Service.

Entwurf

(1) Welche Komponenten werden zur Ausführung eines BPEL-Prozesses benötigt? Welche Komponenten werden zur Behandlung von WS-Security zusätzlich benötigt? Siehe hierzu Information 18, vergleiche auch mit Information 8.

Lösungshinweise:

a) BPEL-Engine als ausführende Einheit, eventuell noch eine SOAP-Engine.
b) WS-Security-fähigen Protocol Handler, eventuell zusätzlich benötigte Artefakte sind Authentication Tokens oder Schlüssel bzw. Zertifikate.

(2) Erläutere Unterschiede zwischen Gateways und Agents!

Lösungshinweise:

Siehe Erläuterungen zu Information 23.

(3) Betrachte die Architektur des JBoss AS, welche Komponenten werden zur Behandlung von Webservices benötigt?

Lösungshinweise:

Siehe hierzu Information 29 und Information 34.

(4) Welche Komponenten werden zur Behandlung von WS-Security zusätzlich benötigt?

Lösungshinweise:

Siehe hierzu Information 35.

Implementierung

(1) Welche Vorteile können von der Migration der AXIS-Webservices auf den JBoss AS erwartet werden?

Lösungshinweis:

Vergleiche hierzu auch Information 40 mit Information 43.

(2) Wie und warum werden Schlüssel verteilt? Welche Schlüssel werden zur Verschlüsselung von Nachrichten verwendet?

Lösungshinweis:

a) Siehe zur Verteilung Information 46: Key Distribution, die Schlüssel warden hier von Hand so verteilt.
b) Zur Verschlüsselung werden symmetrische Schlüssel verwendet, diese werden zunächst selbst mit dem öffentlichen Schlüssel des Empfängers verschlüsselt und übertragen.

(3) Wie ändert sich die Schlüsselverteilung wenn ein weiterer Basis-Webservice hinzukommt (ein weiterer BPEL-Prozess die Dienste nutzen möchte)?

Lösungshinweis:

Der eigene öffentliche Schlüssel wird and den BPEL-Prozess übergeben, und der öffentliche Schlüssel des BPEL-Prozesses wird importiert. Wird allerdings ein neuer BPEL-Prozess hinzugenommen, entsteht ein Problem: der neue BPEL-Prozess benötigt dasselbe Schlüsselpaar, das der alte auch besitzt.

Abkürzungen und Glossar

Abkürzung oder Begriff	Langbezeichnung und/oder Begriffserklärung
AXIS	*Apache eXtensible Interaction System* Apache AXIS ist eine SOAP-Engine zur Entwicklung und zum Betrieb von Webservices und entsprechenden Client-Anwendungen.
BPEL	*Business Process Execution Language* In computer science, the Business Process Execution Language (BPEL) is an XML language to describe business processes. A BPEL program is invoked as a Web service, and it can interact with the external world only by calling Web services. The standard that defines how BPEL is used in Web service transactions is BPEL4WS also known as WS-BPEL. BPEL is designed by IBM and Microsoft, based on their respective work on WSFL and XLANG, which are both superseded by BPEL. In April 2003, BPEL was submitted to OASIS and is now being standardized in the Web Services BPEL Technical Committee [Wikipedia, http://en.wikipedia.org/wiki/BPEL].
Confidentiality	The property that information is not made available or disclosed to unauthorized individuals, entities, or processes [RFC2828].
Integrity	The property that data has not been changed, destroyed, or lost in an unauthorized or accidental manner [RFC2828].
ECTS	*European Credit Transfer System* ist ein *Credit*-System zur Anrechnung von Studienleistungen, welches im Rahmen des Bologna-Prozesses eingesetzt wird. Das System stellt eine Methode zur Messung und zum Vergleich von Studienleistungen bereit und ermöglicht so ihre Übertragung europaweit von Hochschule zu Hochschule. Transparenz der akademischen Lehrangebote wird durch die Bereitstellung von detaillierten Informationen über die jeweiligen Studiengänge und den Stellenwert der einzelnen Lehrveranstaltungen erreicht. Wichtigstes Element von *ECTS* ist u.a. ein Notenauszug, der an allen europäischen Hochschulen anerkannt wird, der so genannte *Transcript of Records* (ToR). Er enthält neben den üblichen Daten zur Person sowie zur Heimat- und Gasthochschule die Kursnummer, den Titel, die Dauer, die Benotung sowie die vergebenen *ECTS-Credits* je Modul. Bei der Realisierung dieses Systems kommen Verschiedene Technologien zum Einsatz. Diese sind als Vorgabe zu betrachten, weil die Services bereits laufen.
HTTP	*HyperText Transfer Protocol* Application protocol used in the World Wide Web.
J2EE	*Java-2-Platform Enterprise Edition* Variante eines Java-Rahmenwerks, bei dem der Schwerpunkt auf der Entwicklung von Server-Applikationen (mittels Java Beans) liegt.

JAAS	*Java Authentication and Authorization Service* ist API, die Teil des von J2EE bereitgestellten einheitlichen Sicherheitsmodells ist.
JBoss	JBoss Inc. ist eine Firma, deren Hauptprojekt die Implementierung eines J2EE-konformen Applikationsservers (JBoss AS) ist.
JBoss AS	*JBoss Application Server* ist ein Open-Source-Applicationserver, der von der Firma JBoss entwickelt wird.
JBossCX	*JBoss Connector Service* Bietet einen Connection-Pool für JBoss AS zu Verbindungen mit JCA-Ressourcenadaptern an.
JBossMQ	*JBoss Messaging* ist eine Implementierung der JMS für JBoss AS.
JBossNS	*JBoss Naming Service* ist eine Implementierung nach JNDI-J2EE-Spezifikation für die Lokalisierung der Objekte und Ressourcen für JBoss AS.
JBossSX	*JBoss Security Service* ist eine Sicherheitsimplementierung des JBoss AS.
JBossTX	*JBoss Transaction Service* ist ein von JTA/JTS gestützter Transaktionsmonitor für JBoss AS.
JBossWS	JBoss Infrastrukturkomponente zur Bereitstellung von Webservices, enthält Webservice-Bibliotheken, SOAP-Engine und Entwicklungs-Werkzeuge.
JCA	*J2EE Connector Architecture* ist eine API zur Integration der heterogenen J2EE-Anwendungen.
JCo	*SAP Java Connector* A Java-based middleware, acting as a bridge between ABAP and Java.
JMS	*Java Messaging Service* ist eine Java-API für den Nachrichtenaustausch zwischen Klienten.
JMX	*Java Management eXtension* ist eine Spezifikation zur Verwaltung und Überwachung von Java-

	Anwendungen.
JDBC	*Java Database Connectivity* API, die in Java-Programmen zur Unterstützung des Zugangs zu Datenbanken genutzt wird.
Legacy System	A legacy system is an antiquated computer system or application program which continues to be used because the user (typically an organization) does not want to replace or redesign it.
OWSM	*Oracle Web Services Manager*
PEP	*Policy Enforcement Point* PEPs enforce security policies at runtime by referring to policy decision points (PDPs). PEPs may be combined with PDPs for performance reasons, or as a result of product packaging [BG-GLOS]
Policy	A security policy is the set of rules, principles, and practices that determine how security is implemented in an organization. It must maintain the principles of the organization's general security policy.
Privacy	The right of an entity (normally a person), acting in its own behalf, to determine the degree to which it will interact with its environment, including the degree to which the entity is willing to share information about itself with others [RFC2828].
SAP CM	*SAP Campus Management* Software module Campus Management of the SAP R/3 system especially for University Resource Planning (URP).
Security	(1) Measures taken to protect a system. (2) The condition of a system that results from the establishment and maintenance of measures to protect the system. (3) The condition of system resources being free from unauthorized access and from unauthorized or accidental change, destruction, or loss [RFC2828].
SOA	*Service-oriented Architecture* "A Service-oriented Architecture (SOA) is a form of distributed systems architecture that is typically characterized by the following properties: • Logical view: The service is an abstracted, logical view of actual programs, databases, business processes, etc., defined in terms of what it does, typically carrying out a business-level operation. • Message orientation: The service is formally defined in terms of the messages exchanged between provider agents and requester agents, and not the properties of the agents themselves. The internal structure of an agent, including features such as its implementation language, process structure and even database structure, are deliberately abstracted away in the SOA: using the SOA discipline one does not and should not need to know how an agent implementing a service is constructed. A key benefit of this concerns so-called legacy systems.

By avoiding any knowledge of the internal structure of an agent, one can incorporate any software component or application that can be "wrapped" in message handling code that allows it to adhere to the formal service definition.

- Description orientation: A service is described by machine-processable metadata. The description supports the public nature of the SOA: only those details that are exposed to the public and important for the use of the service should be included in the description. The semantics of a service should be documented, either directly or indirectly, by its description.
- Granularity: Services tend to use a small number of operations with relatively large and complex messages.
- Network orientation: Services tend to be oriented toward use over a network, though this is not an absolute requirement.
- Platform neutral: Messages are sent in a platform-neutral, standardized format delivered through the interfaces. XML is the most obvious format that meets this constraint."

[W3C Working Group Note 11.02.2004, http://www.w3.org/TR/ws-arch/]

SOAP *Simple Object Access Protocol*

Acronym was dropped in Version 1.2 of the SOAP specification.
SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses. SOAP can potentially be used in combination with a variety of other protocols; however, the only bindings defined in this document describe how to use SOAP in combination with HTTP and HTTP Extension Framework.
http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

ToR *Transcript of Records*

The Bologna Process aims to establish an European area of higher education converging towards a more transparent system of academic and professional recognition of qualifications. Therefore two major concepts have been introduced:

- the European Credit Transfer System (ECTS) and
- the Transcript of Records (ToR).

ECTS is a scheme that helps to evaluate the performed work of a student (within Europe) in an understandable and comprehensive form. It takes into account the workload of course units in general and examination results of the student to allow a better classification for abroad studies.

A ToR is an extraction which includes all the performed work of a student in ECTS norm at a certain date [We05].

Web Service "A Web service is a software system designated to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact

with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. A Web service is an abstract notion that must be implemented by a concrete agent. The agent is the concrete piece of software or hardware that sends and receives messages, while the (Web-) service is the resource characterized by the abstract set of functionality that is provided."
[W3C Working Group Note 11.02.2004, http://www.w3.org/TR/ws-arch/]

WSDL *Web Service Description Language*

WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate, however, the only bindings described in this document describe how to use WSDL in conjunction with SOAP 1.1, HTTP GET/POST, and MIME [W3C, http://www.w3.org/TR/wsdl].

WUSKAR *Werkstatt Unternehmens Software Karlsruhe*

Das zentrale Ziel des Projekts WUSKAR ist eine praxisnahe Ausbildung für Studierende Karlsruher Hochschulen anzubieten, indem eine Plattform geschaffen wird, mit der die Studierenden lernen, Fragestellungen aus der Praxis mithilfe von Unternehmenssoftware umzusetzen. Dabei handelt es sich um ein solches vom Land gefördertes Projekt mit einer geplanten Laufzeit von drei Jahren.

XACML *eXtensible Access Control Markup Language*

XACML is an acronym for eXtensible Access Control Markup Language. It is a declarative access control policy language implemented in XML. XACML includes a policy language and a query language that results in a Permit, Deny, Intermediate (error in query) or Not Applicable response.

XML *eXtensible Markup Language*

W3C recommendation for creating special-purpose markup languages; it is a simplified subset of SGML, capable of describing many different kinds of data.

XSLT *eXtensible Stylesheet Language Transformations*

is designed for use as part of XSL, which is a stylesheet language for XML. In addition to XSLT, XSL includes an XML vocabulary for specifying formatting. XSL specifies the styling of an XML document by using XSLT to describe how the document is transformed into another XML document that uses the formatting vocabulary [W3C consortium, http://www.w3.org].

Index

„Ende-zu-Ende“-Absicherung....18
Aktivitätsdiagramm....9
Altsysteme (legacy system)....12
Business Process Execution Language (BPEL)....19
contract driven development....57
EJB....32
Gateway....22
Geschäftsobjektdiagramm....8
Handler....22
Interceptor....22
Interoperabilität....56
Java 2 Platform Enterprise Edition (J2EE)....29
Java Connector (JCo)....40
JBoss AS....32
JBossWS....38
Middleware Layer....22
Oracle Application Server....24
Oracle BPEL Process Manager....25
Oracle Container for Java (OC4J)....25
Oracle Process Manager and Notification Server (OPMN)....24
Oracle SOA Suite....24
Oracle Web Services Manager (OWSM)....27
Plain Old Java Object (POJO)....32
SAP CM....40
SAP R/3....12
Secure Socket Layer (SSL)....15
Servlet....32
Transcript of Records (ToR)....7
Web Service Invocation Layer (WSIF)....26
Werkstatt Unternehmenssoftware Karlsruhe (WUSKAR)....4
Wrapper....12
WS-Security....16
XML Encryption....15
XML-Signature....15
Zugriffskontrolle....22

Informationsfolien

Information 1: WUSKAR CASE STUDY - Goals....1
Information 2: WUSKAR CASE STUDY - Content Overview....6
Information 3: Use Cases including the Transcript of Records Service....7
Information 4: Decomposed Use Case: Transcript of Records Service....8
Information 5: Business Object Diagram: Transcript of Records....9
Information 6: Activity Diagram of Use Cases: Transcript of Records Service....9
Information 7: Threat Scenario: Transcript of Records Service....10
Information 8: ToR BPEL Process interacting with Web Service Components....11
Information 9: Existing Systems....12
Information 10: Business Object Model: Transcript of Records....13
Information 11: Security Requirements....14
Information 12: Security Technologies....15
Information 13: Outlook: WS-Security and Related Specifications....16
Information 14: Securing exchanged Messages within the Transcript of Records Service....18
Information 15: ANALYSIS - Exercises....19
Information 16: System Restrictions....20
Information 17: Web Service Interaction Levels....21
Information 18: Architecture Overview - Middleware Layer....21
Information 19: Transcript of Records Service with Access Control Enforcement....23
Information 20: Oracle SOA Suite - Basic Installation Topology....24
Information 21: Oracle BPEL Process Manager Architecture....25
Information 22: Securing BPEL Processes with Oracle BPEL Process Manager....26
Information 23: Securing Web Services with Oracle Web Services Manager....27
Information 24: Signature and Encryption with OWSM....28
Information 25: Exemplary OWSM Security Policy....28
Information 26: JBoss Application Server and Legacy Systems - Content Overview....29
Information 27: J2EE Architecture Overview....30
Information 28: J2EE Component Models Supported by JBoss AS....32
Information 29: JBoss AS Architecture....33

Information 30: JMX - Layered Model 34
Information 31: Components Overview of JMX Instrumentation Layer 35
Information 32: Components Overview of JMX Agent Layer 36
Information 33: JBoss Services Overview 37
Information 34: JBossWS and JBoss AS 38
Information 35: JBossWS and WS-Security 39
Information 36: Signature and Encryption with JBossWS 39
Information 37: Interaction JBoss AS and SAP CM 40
Information 38: DESIGN - Exercises 41
Information 39: Systems Overview 42
Information 40: Existing StudentDBService 43
Information 41: AXIS Proprietary Artifacts 43
Information 42: AXIS Proprietary Code Fragments 44
Information 43: StudentDBService JSR-181 POJO Endpoint 45
Information 44: Listing 1 - web.xml 45
Information 45: Requirements 46
Information 46: Key Distribution 47
Information 47: Oracle Web Services Manager Control Console 47
Information 48: OWSM Control Console - Add an OWSM Gateway Component 48
Information 49: OWSM Control Console - Register Services 49
Information 50: OWSM Control Console - Pipeline Templates 1 50
Information 51: OWSM Control Console - Pipeline Templates 2 51
Information 52: Securing Web Service Endpoints on JBoss AS 52
Information 53: Listing 2 - web.xml 52
Information 54: Listing 3 - jboss-wsse-server.xml 53
Information 55: Create Keystore 53
Information 56: Create Truststore 54
Information 57: SOAP Message 54
Information 58: IMPLEMENTATION - Exercises 55
Information 59: Interoperability Issues – Exchanged Message 56

Literatur

[AE+04] Sebastian Abeck, Christian Emig, Jochen Weisser: Fallstudie Transcript of Records, Universität Karlsruhe (TH), C&M (Prof. Abeck), 2004.

[Al03] Ian Alexander: Misuse Cases – Use Cases with Hostile Intent, IEEE Software, 2003.

[An06] Anne Anderson: Web Services Policies, Article, IEEE Security & Privacy, 2006.

[Ar04] Ali Arsanjani: Service-oriented modeling and architecture, article, IBM developerWorks, 2004.

[C&M-I-A] Cooperation & Management: ARCHITECTURE OF IT-SYSTEMS, Kurseinheit zur Vorlesung INTERNET-SYSTEME UND WEB-APPLIKATIONEN (ISWA), Universität Karlsruhe (TH), C&M Prof. Abeck, WiSe 2006/07.

[C&M-I-AE] Cooperation & Management: ANWENDUNGSEINFÜHRUNG, Kurseinheit zu INTERNET-SYSTEME UND WEB-APPLIKATIONEN (ISWA), Universität Karlsruhe, C&M (Prof. Abeck), WiSe 2006/07.

[C&M-I-ID] Cooperation & Management: IDENTITY MANAGEMENT IN THE FOCUS OF APPLICATION INTEGRATION, Course Unit of the Lecture INTERNET SYSTEMS AND WEB APPLICATIONS (ISWA), Universität Karlsruhe (TH), C&M (Prof. Abeck), SoSe 2006.

[C&M-Tech-JBoss] Sebastian Kreuzer: Webservices entwickeln und betreiben mit dem JBoss Applcation Server, Studienarbeit (TDoc), Universität Karlsruhe (TH), C&M (Prof. Abeck), September 2006.

[C&M-Tech-WS-Security] Sebastian Kreuzer: Einführung in die Webservice-Absicherung mit WS-Security, Studienarbeit (TDoc), Universität Karlsruhe (TH), C&M (Prof. Abeck), September 2006.

[ES+06] Christian Emig, Heiko Schandua, Sebastian Abeck: SOA-aware Authorization Control, International Conference Software Engineering Advances ICSEA'06, Tahiti / French Polynesia, ISBN 0-7695-2703-5, November 2006.

[EW+06] Christian Emig, Jochen Weisser, Sebastian Abeck: Development of SOA-Based Software Systems – an Evolutionary Programming Approach, International Conference on Internet and Web Applications and Services ICIW'06, Guadeloupe / French Caribbean, ISBN 0-7695-2522-9, February 2006. Best Paper Award.

[Ha05] J. Jeffrey Hanson, Multi-tiered Service-Oriented Systems with JBoss, Oktober 2005, http://www.novell.com/coolsolutions/feature/16192.html.

[Ja03] Doris Janssen, Kopplung von Web Services, Fraunhofer IAO, Stuttgart, Dezember 2003.

[JBOSS-AS-Clustering] JBoss: JBoss Application Server 4 –Clustering Guide, Documentation, JBoss Documentation Library, November 2006.

[JBOSS-AS-Install] JBoss: JBoss Application Server 4 –Installation Guide, Documentation, JBoss Documentation Library, October 2006.

[JBOSS-AS-J2EE] JBoss: JBoss Application Server 4 –J2EE Reference, Documentation, JBoss Documentation Library, November 2006.

[JBOSS-WS-User] JBoss: JBoss Web Service -User Guide, Documentation, JBoss Documentation Library, November 2006.

[Oe04] Bernd Oestereich: Objektorientierte Softwareentwicklung – Analyse und Design mit der UML 2.0, Oldenbourg Verlag München Wien, 2004.

[OMG-QoS] Object Management Group: UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms, OMG Specification, September 2004.

[ORACLE-AS-Install] Oracle: Oracle Application Server –Installation Guide, Documentation, Oracle Documentation Library, December 2006.

[ORACLE-BPEL-Admin] Oracle: Oracle BPEL Process Manager – Administrator's Guide, Documentation, Oracle Documentation Library, December 2006.

[ORACLE-WSM-Admin] Oracle: Oracle Web Services Manager – Administrator's Guide, Documentation, Oracle Documentation Library, December 2006.

[ORACLE-WSM-Deploy] Oracle: Oracle Web Services Manager – Deployment Guide, Documentation, Oracle Documentation Library, December 2006.

[OT05] Sven Overhage, Peter Thomas, WS-Specification: Ein Spezfikationsrahmen zur Beschreibung von Web-Services auf Basis des UDDI-Standards, Universität Augsburg, Technische Universität Darmstadt, 2005.

[Po06] Arjen Poutsma: Spring Web Services – Reference Documentation, 2006.

[RFC2828] Internet Engineering Taskforce (IETF): RFC 2828: Internet Security Glossary, IETF Informational, May 2000, http://www.ietf.org/rfc/rfc2828.txt.

[Ru05] Thies Rubarth, Web Service Security, Hochschule für angewandte wissenschaften hamburg, SS 2005.

[Sc06] Heiko Schandua: Anpassung bestehender Identitätsmanagement-Lösungen an serviceorientierte Architekturen, Diplomarbeit, Universität Karlsruhe (TH), C&M (Prof. Abeck), 2006.

[SL+03] Andreas Schmietendorf, Jens Lezius, Evgeni Dimitrov, Daniel Reitz, Reiner Dumke, Aktuelle Ansätze für Web Service basierte Integrationslösungen, Arbeitsgruppe Softwaretechnik – Software Measurement Laboratory SMLab, Fakultät Informatik – Institut für verteilter Systeme, Otto-von-Guericke-Universität Magdeburg, 2003.

[SS05] Heiko Schandua, Tomas Stiller: Werkstatt Unternehmenssoftware Karlsruhe (WUSKAR), Case Study University SOA, Team Study Thesis, Universität Karlsruhe (TH), C&M (Prof. Abeck), 2005.

[St04] Christian Stahl, Transformation von BPEL4WS in Petrinetze, Diplomarbeit, Humboldt-Universität zu Berlin, Institut für Informatik, April 2004.